

305-CD-025-002

## **EOSDIS Core System Project**

# **Release B SDPS Ingest Subsystem Design Specification**

March 1996

Hughes Information Technology Systems  
Upper Marlboro, Maryland

# **Release B SDPS Ingest Subsystem Design Specification for the ECS Project**

**March 1996**

Prepared Under Contract NAS5-60000  
CDRL Item #046

## **SUBMITTED BY**

Rick Kochhar /s/

3/25/96

---

Rick Kochhar, Release B CCB Chairman  
EOSDIS Core System Project

Date

**Hughes Information Technology Systems**  
Upper Marlboro, Maryland

This page intentionally left blank.

# Preface

---

This document is one of eighteen comprising the detailed design specifications of the SDPS and CSMS subsystem for Release B of the ECS project. A complete list of the design specification documents is given below. Of particular interest are documents number 305-CD-020, which provides an overview of the subsystems and 305-CD-039, the Data Dictionary, for those reviewing the object models in detail.

The SDPS and CSMS subsystem design specification documents for Release B of the ECS Project include:

305-CD-020	Release B Overview of the SDPS and CSMS Segment System Design Specification
305-CD-021	Release B SDPS Client Subsystem Design Specification
305-CD-022	Release B SDPS Interoperability Subsystem Design Specification
305-CD-023	Release B SDPS Data Management Subsystem Design Specification
305-CD-024	Release B SDPS Data Server Subsystem Design Specification
305-CD-025	Release B SDPS Ingest Subsystem Design Specification
305-CD-026	Release B SDPS Planning Subsystem Design Specification
305-CD-027	Release B SDPS Data Processing Subsystem Design Specification
305-CD-028	Release B CSMS Segment Communications Subsystem Design Specification
305-CD-029	Release B CSMS Segment Systems Management Subsystem Design Specification
305-CD-030	Release B GSFC Distributed Active Archive Center Design Specification
305-CD-031	Release B LaRC Distributed Active Archive Center Design Specification
305-CD-033	Release B EDC Distributed Active Archive Center Design Specification
305-CD-034	Release B ASF Data Center Distributed Active Archive Center Design Specification
305-CD-035	Release B NSIDC Distributed Active Archive Center Design Specification
305-CD-036	Release B JPL Distributed Active Archive Center Design Specification
305-CD-037	Release B ORNL Distributed Active Archive Center Design Specification
305-CD-038	Release B System Monitoring and Coordination Center Design Specification
305-CD-039	Release B Data Dictionary for Subsystem Design Specification

Object models presented in this document have been exported directly from CASE or DBMS tools and in some cases contain too much detail to be easily readable within hard copy page constraints. The reader is encouraged to view these drawings on line using the Portable Document Format (PDF) electronic copy available via the ECS Data Handling System (EDHS) at: URL <http://edhs1.gsfc.nasa.gov>.

This document is a formal contract deliverable with an approval code of 2; as such it requires Government review and approval prior to acceptance and use. This document is under ECS contractor configuration control. Once this document is approved, Contractor approved changes are handled in accordance with Class I and Class II change control requirements described in the EOS Configuration Management Plan, and changes to this document shall be made by document change notice (DCN) or by complete revision.

Any questions should be addressed to:

Data Management Office  
The ECS Project Office  
Hughes Information Technology Systems  
1616 McCormick Drive  
Upper Marlboro, MD 20774-5372

# Abstract

---

The Ingest Subsystem consists of a collection of hardware and software that supports the ingest of data into ECS repositories. This volume presents the overview and critical design of the Ingest CSCI and Ingest Client HWCI elements that comprise this subsystem.

**Keywords:** Ingest, PDL, CSCI, HWCI, client, host, working, storage, network, polling, hard-media, preprocessing, session, request, metadata, GUI, Level-0, Release-B.

This page intentionally left blank.

# Change Information Page

---

List of Effective Pages			
Page Number		Issue	
Title		Submitted as Final	
iii through xiv		Submitted as Final	
1-1 and 1-2		Submitted as Final	
2-1 through 2-4		Submitted as Final	
3-1 through 3-24		Submitted as Final	
4-1 through 4-158		Submitted as Final	
5-1 through 5-12		Submitted as Final	
A-1 through A-18		Submitted as Final	
B-1 through B-42		Submitted as Final	
C-1 through C-4		Submitted as Final	
AB-1 through AB-6		Submitted as Final	
GL-1 through GL-12		Submitted as Final	
Document History			
Document Number	Status/Issue	Publication Date	CCR Number
305-CD-025-001	Preliminary	October 1995	95-0765
305-CD-025-002	Submitted as Final	March 1996	96-0230

# Contents

---

## 1. Introduction

1.1	Identification .....	1-1
1.2	Scope .....	1-1
1.3	Document Organization .....	1-1
1.4	Status and Schedule .....	1-2

## 2. Related Documentation

## 3. Ingest Subsystem Overview

3.1	Introduction and Context .....	3-1
3.1.1	Ingest Subsystem Context Diagram .....	3-2
3.2	Ingest Subsystem Overview .....	3-2
3.2.1	Ingest Subsystem Configuration Item (CI) List .....	3-2
3.2.2	Ingest Subsystem Design Rationale .....	3-2
3.2.3	Ingest Subsystem Design Paradigms .....	3-16
3.2.4	Ingest Subsystem Use of Key Design Mechanisms .....	3-17
3.2.5	Ingest Subsystem Key Design Features .....	3-21

## 4. INGST - Ingest CSCI

4.1	CSCI Overview .....	4-1
4.2	CSCI Context .....	4-2
4.3	Ingest CSCI Object Model.....	4-2
4.3.1	CsGateWay Class .....	4-9
4.3.2	DsCIDescriptor Class .....	4-9
4.3.3	DsCIRequest Class .....	4-10
4.3.4	DsGIParameter Class .....	4-10
4.3.5	DsGIParameterList Class .....	4-11
4.3.6	DsStResourceProvider Class .....	4-11
4.3.7	DsStagingDisk Class .....	4-12
4.3.8	EcPFManagedServer Class .....	4-12
4.3.9	InBOBinMetadata Class .....	4-13
4.3.10	InBOMetadata Class .....	4-14
4.3.11	InDAN Class .....	4-15
4.3.12	InDBAccess Class .....	4-16

4.3.13 InDataPreprocessList Class .....	4-19
4.3.14 InDataPreprocessTask Class .....	4-21
4.3.15 InDataServerInsertionTask Class .....	4-23
4.3.16 InDataTransferTask Class .....	4-25
4.3.17 InDataTypeTemplate Class .....	4-28
4.3.18 InExternalDataProviderInfo Class .....	4-30
4.3.19 InFDFData Class .....	4-31
4.3.20 InFile Class .....	4-32
4.3.21 InFileTypeTemplate Class .....	4-34
4.3.22 InGRIBData Class .....	4-37
4.3.23 InGUISession Class .....	4-38
4.3.24 InGranuleAsync_CB Class .....	4-38
4.3.25 InGranuleAsync_SB Class .....	4-40
4.3.26 InGranuleMessageB Class .....	4-42
4.3.27 InGranuleServer_CB Class .....	4-43
4.3.28 InGranuleServer_SB Class .....	4-43
4.3.29 InHDFMetadata Class .....	4-45
4.3.30 InHistoryLog Class .....	4-46
4.3.31 InIngestMainWindow Class .....	4-47
4.3.32 InInteractiveIngestB Class .....	4-48
4.3.33 InLongDAA Class .....	4-50
4.3.34 InLongDDN Class .....	4-51
4.3.35 InMediaIngest Class .....	4-52
4.3.36 InMessage Class .....	4-54
4.3.37 InMetadata Class .....	4-55
4.3.38 InNextAvailableID Class .....	4-56
4.3.39 InPVMetadata Class .....	4-57
4.3.40 InPollingIngestSession Class .....	4-59
4.3.41 InPollingThreshold Class .....	4-62
4.3.42 InReformatData Class .....	4-62
4.3.43 InRequest Class .....	4-63
4.3.44 InRequestController Class .....	4-69
4.3.45 InRequestFileInfo Class .....	4-71
4.3.46 InRequestManager Class .....	4-72
4.3.47 InRequestManager_C Class .....	4-76
4.3.48 InRequestManager_S Class .....	4-77
4.3.49 InRequestProcessData Class .....	4-78
4.3.50 InRequestProcessHeader Class .....	4-79
4.3.51 InRequestSummaryData Class .....	4-83

4.3.52	InRequestSummaryHeader Class .....	4-86
4.3.53	InSDMetadata Class .....	4-87
4.3.54	InScienceData Class .....	4-88
4.3.55	InServer Class .....	4-88
4.3.56	InServerExtRPC_C Class .....	4-89
4.3.57	InServerExtRPC_S Class .....	4-90
4.3.58	InServerIntRPC_C Class .....	4-91
4.3.59	InServerIntRPC_S Class .....	4-91
4.3.60	InSession Class .....	4-92
4.3.61	InSessionEcsRPC_C Class .....	4-94
4.3.62	InSessionEcsRPC_S Class .....	4-95
4.3.63	InSessionExtRPC_C Class .....	4-96
4.3.64	InSessionExtRPC_S Class .....	4-96
4.3.65	InSessionInfo Class .....	4-97
4.3.66	InSessionIntRPC_C Class .....	4-99
4.3.67	InSessionIntRPC_S Class .....	4-100
4.3.68	InShortDAA Class .....	4-100
4.3.69	InShortDDN Class .....	4-101
4.3.70	InSnowIceData Class .....	4-102
4.3.71	InSourceMCF Class .....	4-103
4.3.72	InTOMSData Class .....	4-105
4.4	Ingest CSCI Dynamic Model .....	4-106
4.4.1	Automated Network Ingest (Get) Scenario .....	4-106
4.4.2	Polling Ingest (Files) Scenario .....	4-110
4.4.3	Polling Ingest (Delivery Record) Scenario .....	4-111
4.4.4	Interactive Ingest Scenario .....	4-112
4.4.5	Hard Media Ingest Scenario .....	4-114
4.4.6	Ingest History Log Viewing Scenario .....	4-116
4.4.7	Operator Ingest Status Monitoring Scenario .....	4-118
4.4.8	Interactive Ingest Operator Status Monitoring Scenario .....	4-119
4.4.9	Operator Request Control Scenario .....	4-121
4.4.10	Preprocessing Scenario .....	4-122
4.5	CSCI Structure .....	4-126
4.5.1	Automatic Network Ingest Interface .....	4-129
4.5.2	Polling Ingest Client Interface CSC .....	4-130
4.5.3	Ingest Request Processing CSC .....	4-131
4.5.4	Ingest Data Transfer CSC .....	4-134
4.5.5	Ingest Data Preprocessing CSC .....	4-135
4.5.6	Operator Ingest Interface CSC .....	4-142

4.5.7	Interactive Ingest Interface CSC .....	4-145
4.5.8	Ingest DBMS CSC .....	4-147
4.5.9	Ingest Administration Data CSC .....	4-147
4.5.10	Peripherals CSC .....	4-147
4.5.11	Viewing Tools CSC .....	4-147
4.5.12	Data Storage Software CSC .....	4-148
4.5.13	Resource Administration CSC .....	4-148
4.5.14	Client Interfaces CSC .....	4-148
4.6	Ingest CSCI Management and Operation .....	4-148
4.6.1	System Management Strategy .....	4-148
4.6.2	Operator Interfaces .....	4-154
4.6.3	Ingest Production Reports .....	4-155
4.6.4	Sustaining Engineering Interface to Data Processing Templates .....	4-157

## **5. ICLHW - Ingest Client HWCI**

5.1	Introduction .....	5-1
5.1.1	HWCI Design Drivers .....	5-2
5.1.2	HWCI Structure .....	5-5
5.1.3	Failover and Recovery Strategy .....	5-11

## **Appendix A. Requirements Trace**

## **Appendix B. Program Design Language (PDL)**

## **Appendix C. Ingest Recovery Analysis**

## **Acronyms and Abbreviations**

## **Figures**

3.1-1	Ingest Subsystem Context Diagram .....	3-3
3.1-2	Ingest Subsystem Hardware Diagram .....	3-11
4.3-1.	In_Ingest_Request_Processing Object Model Diagram .....	4-3
4.3-2.	In_Ingest_Granule Processing Object Model Diagram .....	4-4
4.3-3.	In_Ingest_PreProcessing Object Model Diagram .....	4-5
4.3-4.	In_Ingest_Database_Object Model Diagram .....	4-6
4.3-5.	In_Ingest_Session Object Model Diagram .....	4-7
4.3-6.	In_Ingest_Session_Manager Object Model Diagram .....	4-8
4.4-1.	In_Automated_Network_Ingest_Get_Event_Trace_Diagram .....	4-109

4.4-2. In_Polling_Files_Ingest_Event_Trace Dynamic Model .....	4-111
4.4-3. In_Polling_Delivery_Record_Ingest_Event Trace Diagram .....	4-112
4.4-4. In_Interactive_Ingest_Event_Trace_Diagram Dynamic Model .....	4-114
4.4-5. In_Hard_Media_Ingest_Event_Trace_Diagram Dynamic Model .....	4-116
4.4-6. In_Ingest_History_Log_Viewing_Event_Trace_Diagram Dynamic Model .....	4-117
4.4-7. In_Ingest_Operator_Status_Monitoring Event Trace Diagram .....	4-119
4.4-8. In_Interactive_Ingest_Status_Monitoring_Event_Trace Diagram .....	4-120
4.4-9. In_Ingest_Operator_Request_Update_Event_Trace Diagram .....	4-122
4.4-10. In_Ingest_Preprocessing_Scenario1 Event Trace Diagram .....	4-125
4.5-1. Ingest CSC Interaction .....	4-128
4.5-2. Automated Network Ingest Interface .....	4-129
4.5-3. Ingest Polling CSC .....	4-131
4.5-4. Ingest Request Processing CSC .....	4-132
4.5-5. Ingest Preprocessing CSC Data Flow .....	4-136
4.5-6. Media Ingest Capability of the Operator-Ingest Interface CSC.....	4-144
4.5-7 Request Monitoring Capability of the Operator Ingest Interface CSC .....	4-144
4.5-8 Request Control Capability of Operator Ingest Interface CSC .....	4-144
4.5-9 Interactive Ingest Interface CSC .....	4-146
4.5-10 Sample ECS User Ingest Directory Structure .....	4-146
5.1-1 Ingest HWCI Block Diagram .....	5-6
5.1-2 Ingest Network Connectivity .....	5-8

## Tables

3.1-1 Ingest Subsystem Interfaces .....	3-4
3.1-2 External Interface Protocols .....	3-13
3.2-2 Ingest Performance and Accountability Parameters .....	3-19
3.2-3 Ingest Tunable Parameters .....	3-22
4.2-1. Ingest CSCI Service Interfaces .....	4-2
4.4-1. Automated Network Ingest Scenario (Get) Event Trace Diagram .....	4-106
4.4-2. Polling Ingest (Files) Event Trace .....	4-110
4.4-3. Polling Ingest (Delivery Record) Event Trace Diagram .....	4-111
4.4-4. Interactive Ingest Event Trace .....	4-113
4.4-5. Hard Media Ingest Event Trace .....	4-115
4.4-6. Ingest History Log Event Trace Diagram .....	4-116
4.4-7. Operator Ingest Status Monitoring Event Trace .....	4-118
4.4-8. User Ingest Status Monitoring Event Trace .....	4-120
4.4-9. Operator Request Update Event Trace .....	4-121
4.4-10. Preprocessing Event Trace .....	4-123
4.5-1. Ingest CSCI Components .....	4-126

4.5-2. Ingest CSC Component to Class Mappings .....	4-127
4.5-3. Release-A Ingest Data Type Preprocessing Requirements .....	4-138
4.5-4. Release-B Ingest Data Type Preprocessing Requirements .....	4-140
4.6-1. Ingest Subsystem Error Categories .....	4-149
4.6-2. Ingest HTML Forms .....	4-156
4.6-3. Standard Ingest Production Reports .....	4-157
4.6-4. Data Processing Template Stored Procedures .....	4-158
5.1-1 Daily L0 Ingest Volumes .....	5-1
5.1-2 Ingest Client Sizing Factors .....	5-2
5.1-3 Ingest Client Interfaces .....	5-6
5.1-4 Ingest HWCI Interface Drivers .....	5-7
5.1-5 Ingest HWCI Component Descriptions .....	5-8
5.1-6 Annual L0 Storage Volumes .....	5-10
A-1 Requirements Trace .....	A-1
C-1 Ingest Problem Category, Problem Consequences, and Recovery Actions .....	C-1

This page intentionally left blank.

# 1. Introduction

---

## 1.1 Identification

This Ingest Subsystem Design Specification for the ECS Project, Contract Data Requirement List (CDRL) item 046, with requirements specified in data item description (DID) 305/DV2, is a required deliverable under the Earth Observing System Data and Information System (EOSDIS) Core System (ECS), contract NAS5-60000. This publication is part of a series of documents comprising the Science and Communications Development Office design specification for the Communications and System Management Segment (CSMS) and the Science and Data Processing Segment (SDPS) for Release B.

## 1.2 Scope

The Ingest Subsystem Design Specification defines the Release B detailed design of the Ingest Subsystem. It defines the Ingest Subsystem computer software and hardware architectural design, as well as subsystem design based on Level 2 requirements.

This subsystem is on a formal development track. It is released in and reviewed at the formal Release B Critical Design Review.

This document reflects the February 7, 1996 Technical Baseline maintained by the contractor configuration control board in accordance with ECS Technical Direction No. 11 dated December 6, 1994 submitted via contract correspondence number ECS 194-00343.

## 1.3 Document Organization

The document is organized to describe the Ingest Subsystem design as follows:

Section 1 provides information regarding the identification, scope, status, and organization of this document.

Section 2 provides a listing of the related documents, which were used as source information for this document.

Section 3 provides an overview of the Subsystem, focusing on the high-level design concept. This provides general background information to put Ingest into context.

Section 4 contains the structure of the computer software configuration item (CSCI) comprising the Ingest Subsystem. Included are CSCI context diagrams, the CSCI object model, the CSCI dynamic model (scenarios), and the CSCI physical model (executables).

Section 5 contains the hardware configuration item (HWCI) design of the Ingest Subsystem.

Appendix A provides the Level 4 requirements-to-design mapping matrix for use in verifying coverage of the Level 4 requirements.

Appendix B contains Program Design Language (PDL) for the non-trivial object class methods described in Section 4.

Appendix C contains a table of major Ingest subsystem faults and the actions taken to recover.

Appendix D contains transaction load analysis information that was used to size Ingest data bases. An Acronym list and Glossary are provided.

## **1.4 Status and Schedule**

This submittal of DID 305/DV2 meets the milestone specified in the CDRL of NASA contract NAS5-60000. The submittal was originally reviewed during the SDPS Preliminary Design Review (PDR) and reflects changes to the design which resulted from that review. The submittal was subsequently reviewed during the SDPS/CSMS Interim Design Review (IDR) and reflects changes to the design which resulted from that review. The IDR also triggered a number of follow up actions in response to Review Item Discrepancies (RID), the results of which are incorporated into the Release B Critical Design Review (CDR) version of this document.

## 2. Related Documentation

---

### 2.1 Parent Documents

The parent document is the document from which the scope and content of this Ingest Subsystem Design Specification is derived.

194-207-SE1-001	System Design Specification for the ECS Project
305-CD-002-002	Science and Data processing Segment (SDPS) Design Specification for the ECS Project

### 2.2 Applicable Documents

The following documents are referenced within this Subsystem Design Specification, or are directly applicable, or contain policies or other directive matters that are binding upon the content of this document.

209-CD-001-003	Interface Control Document Between EOSDIS Core System (ECS) and the NASA Science Internet
209-CD-002-003	Interface Control Document Between EOSDIS Core System (ECS) and ASTER Ground Data System
209-CD-005-005	Interface Control Document Between EOSDIS Core System (ECS) and Science Computing Facilities (SCF)
209-CD-006-005	Interface Control Document Between EOSDIS Core System (ECS) and National Oceanic and Atmospheric Administration (NOAA) Affiliated Data Center (ADC)
209-CD-007-003	Interface Control Document Between EOSDIS Core System (ECS) and TRMM Science Data and Information System (TSDIS)
209-CD-008-004	Interface Control Document Between EOSDIS Core System (ECS) and the Goddard Space Flight Center (GSFC) Distributed Active Archive Center (DAAC)
209-CD-010-002	Interface Control Document Between EOSDIS Core System (ECS) and the Langley Research Center (LaRC) Distributed Active Archive Center (DAAC)
209-CD-011-003	Interface Control Document Between EOSDIS Core System (ECS) and the Version 0 System
209-CD-013-003	Interface Control Document Between EOSDIS Core System (ECS) and Landsat-7
209-CD-021-002	Interface Control Document Between EOSDIS Core System (ECS) and the Alaska SAR Facility (ASF) Distributed Active Archive Center (DAAC)

209-CD-022-002	Interface Control Document Between EOSDIS Core System (ECS) and the Oak Ridge National Laboratory (ORNL) Distributed Active Archive Center (DAAC)
209-CD-027-001	Interface Control Document Between EOSDIS Core System (ECS) and SAGE-III
305-CD-003-002	Communications and System Management Segment (CSMS) Design Specification for the ECS Project
305-CD-004-001	Overview of Release A System Design Specification
305-CD-009-001	Release A SDPS Ingest Subsystem Design Specification
305-CD-020-002	Release B Overview of the SDPS and CSMS Segment Design Specification
305-CD-025-001	Release B SDPS Ingest Subsystem Design Specification for the ECS Project
305-CD-028-002	Release B CSMS Segment Communications Subsystem Design Specification (IDR Version)
305-CD-030-002	Release B GSFC DAAC Design Specification
305-CD-031-002	Release B LaRC DAAC Design Specification
305-CD-033-002	Release B EDC DAAC Design Specification
305-CD-034-002	Release B ASF DAAC Design Specification
305-CD-035-002	Release B NSIDC DAAC Design Specification
305-CD-036-002	Release B JPL DAAC Design Specification
305-CD-037-002	Release B ORNL DAAC Design Specification
308-CD-001-005	Software Development Plan for the ECS Project
313-CD-004-001	Release A Communications and System Management Segment (CSMS) and Science Data Processing Segment (SDPS) Internal Interface Control Document for the ECS Project
313-CD-006-002	Release B SDPS/CSMS Internal Interface Control Document
420-TP-010-001	Transition to Release B (Technical Paper)
423-41-03	Goddard Space Flight Center, EOSDIS Core System (ECS) Contract Data Requirements Document
515-CD-002-002	Release B Availability Models/Predictions for the ECS Project
605-CD-002-001	Release B SDPS/CSMS Operations Scenarios

## 2.3 Information Documents Not Referenced

The following documents, although not referenced herein and/or not directly applicable, do amplify and clarify the information presented in this document. These documents are not binding on the content of this Subsystem Design Specifications.

205-CD-002-002	Science User's Guide and Operations Procedure Handbook for the ECS Project, Part 4: Software Developer's Guide to Preparation, Delivery, Integration and Test with ECS
206-CD-001-002	Version 0 Analysis Report for the ECS Project
302-CD-003-001	Release B Facilities Plan for the ECS Project
101-303-DV1-001	Individual Facility Requirements for the ECS Project
194-317-DV1-001	Prototyping and Studies Plan for the ECS Project
318-CD-000-XXX	Prototyping and Studies Progress Report for the ECS Project (monthly)
333-CD-003-002	SDP Toolkit Users Guide for the ECS Project
601-CD-001-004	Maintenance and Operations Management Plan for the ECS Project
604-CD-001-004	Operations Concept for the ECS Project: Part 1 -- ECS Overview
604-CD-002-002	Operations Concept for the ECS Project: Part 2B -- Release B
604-CD-003-002	Operations Concept for the ECS Project: Part 2A -- ECS Release A
101-620-OP2-001	List of Recommended Maintenance Equipment for the ECS Project
194-703-PP1-001	System Design Review (SDR) Presentation Package for the ECS Project
194-813-SI4-002	Planning and Scheduling Prototype Results Report for the ECS Project
194-813-SI4-003	DADS Prototype One FSMS Product Operational Evaluation [for the ECS Project]
194-813-SI4-004	DADS Prototype One STK Wolfcreek 9360 Automated Cartridge System Hardware Characterization Report [for the ECS Project]
813-RD-009-001	DADS Prototype Two Multi-FSMS Product Integration Evaluation [for the ECS Project]
828-RD-001-002	Government Furnished Property for the ECS Project
160-TP-002-001	Version 1 Data Migration Plan [for the ECS Project] White Paper
193-WP-118-001	Algorithm Integration and Test Issues for the ECS Project
193-WP-611-001	Science-based System Architecture Drivers for the ECS Project, Revision 1.0
193-WP-623-001	ECS Evolutionary Development White Paper
194-TP-548-001	User Scenario Functional Analysis [for the ECS Project]
194-TP-569-001	PDPS Prototyping at ECS Science and Technology Laboratory, Progress Report #4
194-WP-901-002	EOSDIS Core System Science Information Architecture, White Paper, Working Paper
194-WP-902-002	ECS Science Requirements Summary, White Paper, Working Paper
194-WP-913-003	User Environment Definition for the ECS Project, White Paper, Working Paper
194-WP-925-001	Science Software Integration and Test, White Paper, Working Paper

222-TP-003-008	Release Plan Content Description for the ECS Project
410-TD-001-002	ECS User Interface Style Guide, Technical Data
420-WP-001-001	Maximizing the Use of COTS Software in the SDPS SDS Software Design [for the ECS Project], White Paper
430-TP-001-001	SDP Toolkit Implementation with Pathfinder SSM/I Precipitation Rate Algorithm [for the ECS Project], Technical Paper
440-TP-001-001	Science Data Server Architecture Study [for the ECS Project]
440-TP-014-001	ECS Ingest Subsystem Topology Analysis Technical Paper
423-16-01	Goddard Space Flight Center, Data Production Software and Science Computing Facility (SCF) Standards and Guidelines
423-16-02	Goddard Space Flight Center, PGS Toolkit Requirements Specification for the ECS Project
423-41-02	Goddard Space Flight Center, Functional and Performance Requirements Specification for the Earth Observing System Data and Information System (EOSDIS) Core System
540-022	Goddard Space Flight Center, Earth Observing System (EOS) Communications (Ecom) System Design Specification
560-EDOS-0211.0001	Goddard Space Flight Center, Interface Requirements Document Between EDOS and the EOS Ground System (EGS)

## 3. Ingest Subsystem Overview

---

### 3.1 Introduction and Context

The Ingest Subsystem contains a collection of hardware and software that supports the ingest of data into ECS repositories on a routine and ad hoc basis and triggers subsequent archiving and/or processing of the data. The Ingest Subsystem configuration must be flexible to support a variety of data formats and structures, external interfaces, and ad-hoc ingest tasks. Data processing and storage functions to be performed by the Ingest Subsystem and ingest clients vary according to attributes of the ingested data such as data type, data format, and the level to which the ingested data has been processed.

From a software perspective, the Ingest Subsystem is organized into a collection of tools from which those required for a specific situation can be configured. The resultant configuration is called an ingest client. Ingest clients may exist in a static configuration to service a routine external interface, or they may be specially configured and exist only for the duration of a specific ad hoc ingest task. The ingest clients provide a single virtual interface point for the receipt of all external data to be archived within the SDPS. Individual ingest clients are established to support each unique interface, allowing the interface parameters to be modified as interface and mission requirements evolve. Ingest data preprocessing, metadata extraction, and metadata validation is performed by the ingest clients on any incoming data, as required.

Data is staged to one of two areas depending on the data level, data type, and other data set specific characteristics:

- Level 0 (L0) data from ongoing missions are staged to the highly-reliable Ingest Subsystem working storage area. Metadata is extracted and quality checked on the working storage area. The staged data are accessible by the Planning and Data Processing Subsystem which processes the L0 data into higher-level products. The L0 data are transferred to an archive data repository in the Ingest or Data Server Subsystem for long-term storage.
- Non-L0 data (such as ancillary data and L1a-L4 data from external facilities) are staged directly to the working storage area in the Data Server Subsystem. Extraction of metadata and quality checking are performed on this data upon the Data Server Subsystem processor hardware. Data that are required to support ECS data processing are accessible by the Planning and Data Processing Subsystem from working storage. The non-L0 data are transferred to a Data Server Subsystem archive data repository for long-term storage.

The hardware components of the Ingest Subsystem are similar to those of the Data Server Subsystem, but are specialized to meet the ingest requirements at a given site. Specialized forms of ingest clients may be incorporated into site unique architectures, and additional processing hardware may also be incorporated at those sites where special transformations must be accomplished on ingest data sets.

### 3.1.1 Ingest Subsystem Context Diagram

The Ingest Subsystem must be capable of accepting data from a variety of sources including both electronic network interfaces and hard media. Early interface testing is performed at Interim Release-1 (IR-1) for interfaces at the Sensor Data Processing Facility (SDPF), the Tropical Rainfall Measuring Mission (TRMM) Science Data Information System (TSDIS), and the NOAA Affiliated Data Centers (ADCs). The NOAA ADCs include the National Environmental Satellite, Data, and Information Service (NESDIS) and the GSFC Data Assimilation Office (DAO). Release A interfaces include the SDPF, TSDIS, NOAA ADCs (NESDIS and the DAO), the Science Computing Facilities (SCFs) (for algorithm delivery), the Data Server Subsystem (for archiving), science users, clients (operations staff), and Version 0 DAACs and other ECS DAACs. Early interface testing is performed after Release A for the EOS Data and Operations System (EDOS) and Landsat-7 Processing System (LPS). The EDOS and LPS interfaces are fully functional at Release B. Interfaces are added at Release B for the Landsat International Ground Stations (IGS) and Image Assessment System (IAS), ASTER Ground Data System (GDS), Release B Science Computing Facilities (SCFs), Flight Dynamics Facility (FDF), SeaWinds, SAGE III, ALT/RADAR, ACRIM, and DAAC-unique interfaces (primarily at ASF). Additional interfaces are planned to be implemented in future releases. The Ingest Subsystem context diagram is shown in Figure 3.1-1. Specific details on the interfaces are included in Table 3.1-1.

The following assumptions have been made regarding the characteristics of the data to be ingested:

- Receipt of data from external data providers is assumed to be random, although generally clustered about an expected availability time.
- Data received by the Ingest subsystem has been pre-defined within ECS with regard to expected metadata and metadata characteristics; data types, files, and formats; and means of delivery to ECS.
- The data volume and frequency of receipt of received data is consistent with the ECS technical baseline dated February 7, 1996.

## 3.2 Ingest Subsystem Overview

### 3.2.1 Ingest Subsystem Configuration Item (CI) List

The Ingest Subsystem is composed of one Computer System Configuration Item (CSCI) and one Hardware Configuration Item (HWCI):

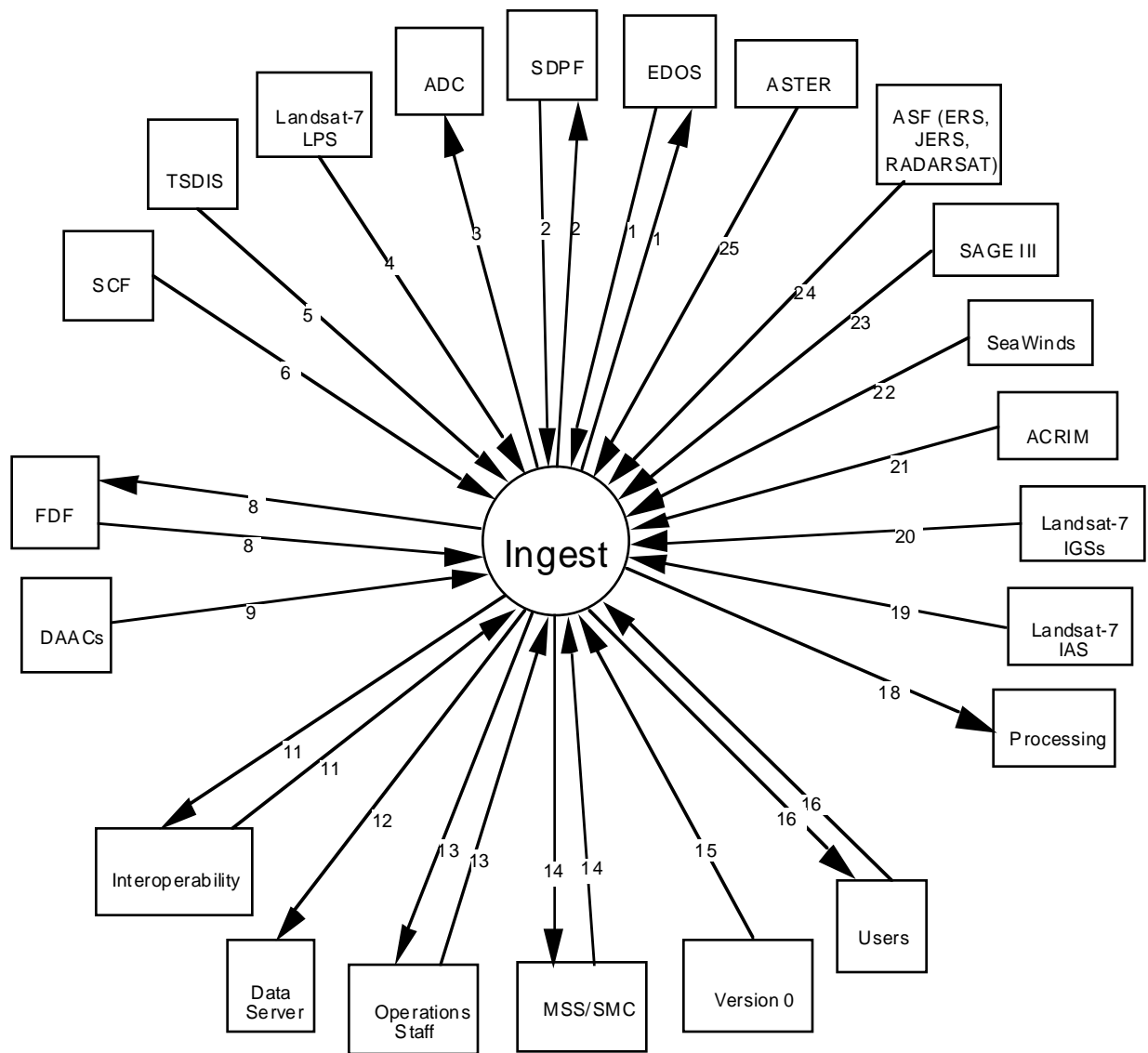
- Ingest CSCI
- Ingest Client HWCI

These CIs are discussed in detail in subsections 4 and 5, respectively.

### 3.2.2 Ingest Subsystem Design Rationale

Main drivers for the design are:

- the high reliability required for Level 0 data ingest
- the required extensibility of the ingest client implementation to future external interfaces



**Figure 3.1-1. Ingest Subsystem Context Diagram**

**Table 3.1-1. Ingest Subsystem Interfaces (1 of 5)**

Flow No.	Source	Destination	Data Types	Data Volume	Frequency
13	Ingest	Operations Staff	Ingest Status	less than 1 MB	in response to request
13	Ingest	Operations Staff	Ingest Threshold Update Status	less than 1 MB	in response to request
13	Ingest	Operations Staff	Ingest Log	less than 1 MB	in response to request
12	Ingest	Data Server	Standard Products	greater than 1 GB	as required for archiving
12	Ingest	Data Server	Metadata	between 1 MB and 1 GB	as required for archiving
12	Ingest	Data Server	Ancillary Data	greater than 1 GB	as required for archiving
12	Ingest	Data Server	Correlative Data	greater than 1 GB	as required for archiving
12	Ingest	Data Server	Calibration Data	between 1 MB and 1 GB	as required for archiving
12	Ingest	Data Server	Documents	between 1 MB and 1 GB	as required for archiving
12	Ingest	Data Server	Orbit/Attitude Data	between 1 MB and 1 GB	as required for archiving
12	Ingest	Data Server	Algorithms	greater than 1 GB	as required for archiving
12	Ingest	Data Server	Special Products	greater than 1 GB	as required for archiving
12	Ingest	Data Server	L0 Data	greater than 1 GB	as required for archiving
12	Ingest	Data Server	Quick Look Data	between 1 MB and 1 GB	as required for archiving
12	Ingest	Data Server	QA Data	between 1 MB and 1 GB	as required for archiving
12	Ingest	Data Server	Resource Allocation / Deallocation Requests	less than 1 MB	on receipt of data
12	Ingest	Data Server	Data Insert Request	less than 1 MB	on receipt of data
1	Ingest	EDOS	Acceptance Notification File	less than 1 MB	upon completion of data ingest into DSS
8*	Ingest	FDF	Data Delivery Notice	less than 1 MB	upon completion of data insert into DSS
11	Ingest	Interoperability	Search Request	less than 1 MB	as required for archiving
11	Ingest	Interoperability	Advertisement	less than 1 MB	when capability changes

**Table 3.1-1. Ingest Subsystem Interfaces (2 of 5)**

Flow No.	Source	Destination	Data Types	Data Volume	Frequency
11*	Ingest	Interoperability	Subscription	less than 1 MB	in response to request
14	Ingest	MSS/SMC	Ingest status	less than 1 MB	as performance and configuration parameters change in state
14	Ingest	MSS/SMC	Ingest log	less than 1 MB	as reportable events occur
18	Ingest	Processing	L0 data	greater than 1 GB	as requested by Processing for L0 to higher level processing
16	Ingest	Users	Ingest Status	less than 1 MB	as requested
21*	ACRIM	Ingest	L1A data	between 1 MB and 1 GB	TBD
3	ADC (NES-DIS and GDAO)	Ingest	Metadata	less than 1 MB	several times a day
3	ADC (NES-DIS and GDAO)	Ingest	Ancillary Data	between 1 MB and 1 GB	several times a day
3	ADC (NES-DIS and GDAO)	Ingest	Calibration Data, Correlative Data, Documents	less than 1 MB	several times a day
24*	ASF	Ingest	Data Products	greater than 1 GB	several times a day
24*	ASF	Ingest	Browse	between 1 MB and 1 GB	several times a day
24*	ASF	Ingest	Ancillary	between 1 MB and 1 GB	several times a day
24*	ASF	Ingest	Metadata	less than 1 MB	several times a day
25*	ASTER	Ingest	Data Products (L1a, L1b data)	greater than 1 GB	daily
25*	ASTER	Ingest	Ancillary	greater than 1 GB	daily
25*	ASTER	Ingest	Metadata	between 1 MB and 1 GB	daily
25*	ASTER	Ingest	Browse	greater than 1 GB	daily
13	Client Operations Staff	Ingest	ingest status requests	less than 1 MB	as requested
13	Operations Staff	Ingest	ingest log requests	less than 1 MB	as requested

**Table 3.1-1. Ingest Subsystem Interfaces (3 of 5)**

Flow No.	Source	Destination	Data Types	Data Volume	Frequency
13	Operations Staff	Ingest	ingest control requests	less than 1 MB	as requested
13	Operations Staff	Ingest	ingest threshold control requests	less than 1 MB	as requested
9	DAACs	Ingest	Ancillary Data	greater than 1 GB	as required
9	DAACs	Ingest	Correlative Data	greater than 1 GB	as required
9	DAACs	Ingest	Calibration Data	between 1 MB and 1 GB	as required
9	DAACs	Ingest	QA Data	between 1 MB and 1 GB	as required
1	EDOS	Ingest	PDSs (L0 Data)	greater than 1 GB	several times a day
1*	EDOS	Ingest	ADSs (Back-up L0 Data)	greater than 1 GB	infrequently
1	EDOS	Ingest	PDS Delivery Record	less than 1 MB	several times a day
1*	EDOS	Ingest	Physical Media Unit Delivery Record	less than 1 MB	infrequently
8*	FDF	Ingest	Repaired Orbit Data	between 1 MB and 1 GB	infrequently
11	Interoperability	Ingest	Notification	very low	as requested
11	Interoperability	Ingest	Advertisement Info.	less than 1 MB	in response to request
4	Landsat 7 PS	Ingest	Metadata	less than 1 MB	several times a day
4	Landsat 7 PS	Ingest	L0R Science Data	139.4 GB/day	several times a day
4	Landsat 7 PS	Ingest	L0R Browse	between 1 MB and 1 GB	several times a day
4	Landsat 7 PS	Ingest	Data Availability Notice	less than 1 MB	several times a day
4	Landsat 7 MOC	Ingest	Activity Calendar	between 1 MB and 1 GB	TBR
4	Landsat 7 PS	Ingest	Payload Correction Data	between 1 MB and 1 GB	several times a day
4	Landsat 7 PS	Ingest	Mirror Scan Correction Data	between 1 MB and 1 GB	several times a day
4	Landsat 7 PS	Ingest	Calibration data	between 1 MB and 1 GB	several times a day

**Table 3.1-1. Ingest Subsystem Interfaces (4 of 5)**

Flow No.	Source	Destination	Data Types	Data Volume	Frequency
4	Landsat 7 PS	Ingest	Data Availability Notice	less than 1 MB	several times a day
4	Landsat 7 PS	Ingest	Browse Data	greater than 1 GB	as required
4	Landsat 7 PS	Ingest	Directory and Guide Information	between 1 MB and 1 GB	infrequently
20	Landsat 7 IGS	Ingest	Inventory Data	between 1 MB and 1 GB	monthly
20	Landsat 7 IGS	Ingest	Browse Data	greater than 1 GB	monthly
19	Landsat 7 IAS	Ingest	Calibration Data	between 1 MB and 1 GB	infrequently
19	Landsat 7 IAS	Ingest	Metadata	between 1 MB and 1 GB	infrequently
14	MSS/SMC	Ingest	Ingest status requests	less than 1 MB	in response to request
14	MSS/SMC	Ingest	Ingest log requests	less than 1 MB	in response to request
23*	SAGE III	Ingest	L0 Data	between 1 MB and 1 GB	TBD
23*	SAGE III	Ingest	Browse	less than 1 MB	TBD
23*	SAGE III	Ingest	Ancillary	less than 1 MB	TBD
23*	SAGE III	Ingest	Metadata	less than 1 MB	TBD
23*	SAGE III	Ingest	Definitive/predicted orbit data	less than 1 MB	TBD
23*	SAGE III	Ingest	Expedited data	less than 1 MB	TBD
6*	SCF	Ingest	Ancillary, calibration, correlative data	between 1 MB and 1 GB	as required
6*	SCF	Ingest	Metadata	less than 1 MB	as required
6*	SCF	Ingest	Documents	less than 1 MB	as required
6	SCF	Ingest	Algorithms/Updates	between 1 MB and 1 GB	as required
2	SDPF	Ingest	L0 Data	65 MB/day - MSFC 90 MB/day - LaRC	daily
2	SDPF	Ingest	Housekeeping		daily
2	SDPF	Ingest	Quick Look Data	between 1 MB and 1 GB	three times a day

**Table 3.1-1. Ingest Subsystem Interfaces (5 of 5)**

Flow No.	Source	Destination	Data Types	Data Volume	Frequency
2	SDPF	Ingest	Predictive Orbit Data	between 1 MB and 1 GB	daily
2	SDPF	Ingest	Definitive Orbit Data	between 1 MB and 1 GB	daily
2	SDPF	Ingest	Data Availability Notice	less than 1 MB	daily
2	SDPF	Ingest	Back-up Data	between 1 MB and 1 GB	as required
22*	SeaW-inds	Ingest	L0 Data	between 1 MB and 1 GB	TBD
22*	SeaW-inds	Ingest	Definitive/predicted orbit data	between 1 MB and 1 GB	TBD
22*	SeaW-inds	Ingest	Expedited data	between 1 MB and 1 GB	TBD
5	TSDIS	Ingest	Metadata	between 1 MB and 1 GB	several times a day
5	TSDIS	Ingest	Data Availability Notice	less than 1 MB	several times a day
5	TSDIS	Ingest	Data Products	60 (GB/day) Processing and Reprocessing	provided throughout the day
5	TSDIS	Ingest	Algorithms	between 1 MB and 1 GB	as required
5	TSDIS	Ingest	Documents	less than 1 MB	as required
5	TSDIS	Ingest	Browse Data	149 MB/day	several times a day
5	TSDIS	Ingest	Directory	less than 1 MB	as required
5	TSDIS	Ingest	Guide	less than 1 MB	as required
16	Users	Ingest	Ingest Status Requests	less than 1 MB	as required
15	Version 0	Ingest	Migration Data	greater than 1 GB	varies depending on migration strategy

- Items marked with an asterisk (\*) in the Flow No. column are interfaces to be implemented in Release B.
- Ingest uses the Advertisement Information to locate the relevant Data Servers with which it needs to interact.
- In the table, where an exact number is unavailable, the data volume is estimated as low (less than 1 MB), medium (between 1 MB and 1 GB), or high (greater than 1 GB) per use defined in the frequency column. The frequency information will be updated as the interfaces are fully defined.

- the demands which are imposed on ingest by the migration of Version 0 data from DAAC repositories external to ECS; and
- other performance requirements related to data ingest

The Ingest Subsystem design incorporates the following measures in response to the above drivers:

- The need for high reliability to support the function of Level 0 science data ingest was resolved by the logical and physical separation of the Ingest Subsystem (Level 0) working storage instantiation from other ECS working storage. The ingest of Level 0 data has a very high priority, and must be supported with high component reliability and availability. Maintaining this level of reliability, maintainability, and availability (RMA) throughout the entire SDPS would be prohibitively expensive. Separating a high RMA ingest complement of hardware and software from other SDPS functions allows each subsystem within SDPS to support only the level of RMA necessary to perform its required functions.
- The need for future extensibility was resolved by providing a separate ingest processing component with template (i.e., table-driven) interface software that may be reused as new interfaces are added or old interfaces modified. The external interfaces to be supported by the ingest clients change over time as spacecraft and instruments are added and removed. Each external interface must potentially be supported with a different data transfer mechanism, format conversion, quality checking, metadata definition, and other attributes unique to that data. Separating the performance of these functions from the Level 0 data repository component minimizes or eliminates changes to the data server configuration as mission requirements change.
- In addition, each new or modified external interface may require custom interface software to facilitate the data transfer process. The long-term EOS program expects to add large numbers of new interfaces over time. The Ingest Subsystem software is designed in a modular fashion so as to minimize the development effort required for new or modified interfaces.
- The volume and complexity of data provided from the Version 0 facilities to the ECS for archival are critical design drivers for the ECS Ingest Subsystem. Over 600 data products have been identified, ranging in volume from megabytes to hundreds of gigabytes. Total volume is on the order of dozens of terabytes. Many of the data products are stored in some form of Hierarchical Data Format (HDF); however, many more products are stored in other formats. The Ingest Subsystem software is designed to generalize the mechanism by which data is routinely stored within the SDPS, given a set of standalone tools used to prepare Version 0 data. Version 0 "data preparation" includes retrieval from Version 0-specific hard media, conversion to EOS-HDF, where required, and extraction of standard metadata.
- ECS satisfies explicit performance requirements with the design described in Sections 4 and 5 of this volume.

The following subsections elaborate upon the above design drivers.

### **3.2.2.1 Ingest RMA Architecture**

A principal objective of separating ingest from other SDPS functions is to assure the high reliability and availability of the system for Level 0 data ingest. Ingest availability requirements are met through the use of high reliability components in redundant configurations, as necessary. The following paragraphs provide additional detail on ingest RMA requirements and how the ingest architecture ensures that these requirements will be met.

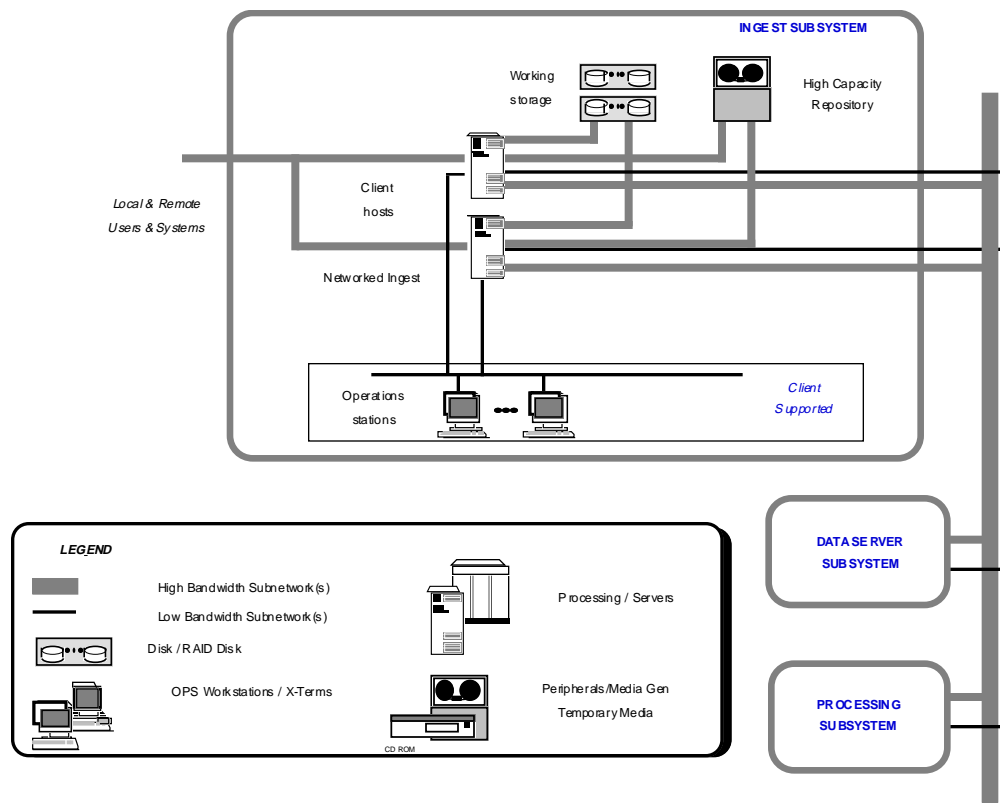
The principal Level 0 data sources (e.g., EDOS, SDPF) each support a data driven architecture that processes data within 24 hours or less from receipt of the data from the spacecraft. Once the data is processed into Production Data Sets (PDSs), the data is transferred to ECS in a timely fashion for archiving and any required higher level processing. Typically, data transfer must be completed within several hours to free up resources at the Level 0 processing sites, as new data sets are being received on a nearly continuous basis. The Level 0 data sources provide long term archiving of the PDSs in the event that data is corrupted or lost in the transfer to ECS or within ECS itself. While it is possible to access these archived data sets if they are needed, it is unattractive to do so from an operational standpoint due to the added time and complexity of pulling the data from the deep archive and staging it for transfer. Three stringent RMA requirements are levied on the Ingest Subsystem to mitigate the need for frequently retrieving data from the Level 0 data providers:

- Ingest Subsystem availability of 0.999
- 2000 hours mean time between failures (MTBF)
- Fifteen minutes mean time to restore (MTTR) from a failure to operational capability

A more detailed availability analysis may be found in the Availability Models/Predictions for the ECS Project document (515-CD-002-002).

It should be noted that certain sites are not staffed 24 hours per day, and are subject to somewhat different operational requirements during unstaffed periods. The Ingest Subsystem is designed to operate with minimal operator involvement, and is planned to continue the function of ingesting data during unstaffed periods. However, certain functions that inherently require initial operator involvement, such as initiating ingest via hard media, are not supported during unstaffed periods. Moreover, faults may require human involvement to reconfigure and reboot the ingest client hosts. Therefore, the 15 minute MTTR requirement for support of the receipt of science data is not be supported during unstaffed periods. In order to eliminate any potential system overload, Ingest Subsystem capabilities are sized to satisfy full 24 hour L0 data ingest requirements during staffed periods.

The instantiation of the Ingest Subsystem varies at each site, but is based on the same architecture concepts and classes of hardware and software. The generic Ingest Subsystem architecture is shown in Figure 3.1-2. The ingest client software required for a specific Level 0 interface at a given site runs on a client host computer residing in the Ingest Subsystem. Multiple ingest clients may run on a single client host, or may be distributed among multiple client hosts, depending on the data load supported by each interface. At least one spare client host is provided at each site in order to provide a warm backup failover capability in the event of a primary host failure. A hot backup approach was initially considered, but was determined to be unnecessary to meet the 15-minute MTTR requirement. Furthermore, a hot backup capability requires the Level 0 data



**Figure 3.1-2. Ingest Subsystem Hardware Diagram**

provider to transfer the same data to two processors (not currently supported by EDOS). Commercially available workstations of the class required for the client hosts typically support MTBFs in the 20,000 to 40,000 hour range. Calculations indicate that the use of this hardware in conjunction with redundant working storage and data repository components will meet data ingest RMA requirements.

Working storage media drive and robotics devices are sized to accommodate redundant devices or components as necessary. A combination of paper analysis and system modeling efforts have been used to determine the final configuration necessary to meet system performance and RMA requirements. A summary of the Ingest Subsystem sizing analysis is presented in the DAAC-specific Design Specifications.

### 3.2.2.2 Ingest Client Implementation

As shown in the context diagram in subsection 3.1.1, the Ingest Subsystem supports a wide variety of external interfaces. The application-level protocol to set up for data transfer is potentially different for each of the external interfaces. As a result, a separate ingest client software application is required to facilitate data transfer for each interface. To minimize the software development effort and make it easier to accommodate new external interfaces in the future, the external interfaces were categorized based on common characteristics as follows:

- Automated Network Ingest by means of a Data Availability Notice (DAN) supplied to ECS--ECS receives the DAN and schedules automated network data transfer from the source. The DAN describes the location of the available data. ECS "gets" data from the source within a specified time window. Note: External data providers are responsible for developing application software to interact with ECS automated network ingest software. The SDPF has existing design/software that may be used as a template.
- Polling Ingest with Delivery Record--ECS periodically checks an agreed-upon network location for a Delivery Record file. The Delivery Record file contains information identical to that in a DAN. The Delivery Record describes the location of the available data. If a Delivery Record is located, ECS "gets" data from the source within a specified time window. Note: the data location may be on a working storage device within ECS, where an external data provider may have previously transferred the data.
- Polling Ingest without Delivery Record--ECS periodically checks an agreed-upon network location for available data. All data in the location is assumed to make up a collection of ingest data of one specific data type, with one file per data granule. If data is located, ECS "gets" data from the source within a system-tunable time window.
- Manual data transfer mechanisms are in place for transfer of data from hard media, for hardcopy scanning and digitizing, and for science user-controlled (interactive) network data transfer.
  - Hard Media Ingest is available for authorized institutions or science users providing data on hard media and as a backup mechanism for facilities where automated network data transfer is temporarily unavailable. The hard media must contain information identical to the Delivery Records described above, in a standard file format, or the data provider must separately provide Delivery Records to a specified ECS location in the standard file format.
  - Hardcopy scanning and digitizing is provided to allow ingest of existing hardcopy documents. Operations personnel digitize/scan the hardcopy. Ingest GUI software is provided to use operator input in building the equivalent of a DAN. Ingest processing proceeds as for other data sources thereafter.
  - Interactive Network Ingest is available for authorized science users to manually identify data to be ingested. Science users may "put" the data into an accessible ECS location or may request that ECS "get" the data from their workstation. Information identical to that contained in the Delivery Record is entered by means of GUI input (or derived by the GUI software).

Data transfer is accomplished by one of three means--file transfer protocol (ftp) "get", ftp "put", or hard media data transfer. Ftp get involves ECS "getting" data from an external site. Ftp put involves an external site "putting" data into ECS. Hard media data transfer involves data transfer from one of several ingest peripheral types found at a DAAC. Kerberized ftp (kftp), providing additional communications security services, is specified as the standard data ingest protocol. Exceptions may be made in the case of interfaces to existing facilities, which may be granted waivers from supporting kftp on a case-by-case basis.

Table 3.1-2 describes each external interface in Table 3.1-1 in terms of the interface protocols used, and the Interface Control Document (ICD) in which detailed interface design information may be found.

**Table 3.1-2. External Interface Protocols (1 of 2)**

<b>Interface (facility)</b>	<b>Type of Primary Interface Protocols</b>	<b>Type of Backup In- terface Protocols</b>	<b>Comments</b>
SDPF	Automated Network Ingest/ ftp get	None	Defined by SDPF ICD
TSDIS	Automated Network Ingest/ ftp get	None	Defined by ECS/TSDIS ICD
NOAA/NMC (DAO)	Polling Ingest without Deliv- ery Record/ftp get	None	Defined by ECS/GSFC DAAC- specific ICD
NOAA (NESDIS)	Polling Ingest without Deliv- ery Record/ftp get	None	Defined by ECS/ADC(NOAA) ICD
EDOS	Polling Ingest with Delivery Record/ftp put	Hard media	Defined by EDOS ICD
ASTER DPS	TBD hard media	None	Defined by ECS/IP ICD
FDF	Automated Network Ingest/ ftp get	None	Defined by ECS/FDF ICD
Landsat7 LPS	Automated Network Ingest/ ftp get	None	Defined by Landsat-7 ICD
Landsat 7 IAS	Interactive Network Ingest/ ftp get or put	None	Defined by Landsat-7 ICD
Landsat 7 IGS	Hard media	None	Defined by Landsat-7 ICD
SCF	Interactive Network Ingest/ ftp get or put	8mm tape	Defined by ECS/SCF ICD
Users	Interactive Network Ingest/ ftp get or put	None	Defined by TBD
Version 0 (GS- FC)	Hard Media Ingest (8mm tape) and/or Polling Ingest with Delivery Record/ftp get and/or Interactive Network Ingest/ftp get or put	None	Defined by ECS/GSFC DAAC- specific ICD
Version 0 (LaRC)	Polling Ingest with Delivery Record/ftp get plus Interac- tive Network Ingest/ftp get or put	None	Defined by ECS/LaRC DAAC- specific ICD
Version 0 (JPL)	Hard Media Ingest (8mm tape) and/or Polling Ingest with Delivery Record/ftp get and/or Interactive Network Ingest/ftp get or put	None	Defined by the ECS/JPL DAAC- specific ICD
Version 0 (NSIDC)	Hard Media Ingest (8mm tape) and/or Polling Ingest with Delivery Record/ftp get and/or Interactive Network Ingest/ftp get or put	None	Defined by the ECS/NSIDC DAAC-specific ICD

**Table 3.1-2. External Interface Protocols (2 of 2)**

Interface (facility)	Type of Primary Interface Protocols	Type of Backup Interface Protocols	Comments
Version 0 (ASF)	Hard Media Ingest (8mm tape) and/or Polling Ingest with Delivery Record/ftp get and/or Interactive Network Ingest/ftp get or put	None	Defined by the ECS/ASF DAAC-specific ICD
Version 0 (EDC)	Hard Media Ingest (8mm tape) and/or Polling Ingest with Delivery Record/ftp get and/or Interactive Network Ingest/ftp get or pu	None	Defined by the ECS/EDC DAAC-specific ICD
Version 0 (CIES-IN)	All available for CIESIN implementation	None	Defined by the ECS/CIESIN DAAC-specific ICD
Version 0 (ORNL)	All available for ORNL implementation	None	Defined by the ECS/ORNL DAAC-specific ICD
SeaWinds	Polling with Delivery Record/ftp get	None	Defined by the ECS/SeaWinds ICD
ALT/RADAR	Polling with Delivery Record/ftp get	None	Defined by the ECS/ALT/RADAR ICD
SAGE III	Polling with Delivery Record/ftp get	None	Defined by the ECS/SAGE III ICD
ACRIM	Polling with Delivery Record/ftp get	None	Defined by the ECS/ACRIM ICD

### 3.2.2.3 Version 0 Migration Impact on the Ingest Subsystem

Requirements that dictate the volume and other characteristics of Version 0 data to be migrated in the Release A and Release B timeframe from the Version 0 DAACs into the ECS are a critical design driver. The Version 0 data widely varies in format, structure, volume, and transfer mechanism. Previous design team experience indicates that migration of existing data requires a major engineering effort, including data analysis; data conversion and reformatting tool development; extensive integration and test (to ensure data integrity); system analysis of required hardware components (e.g., networks, storage, etc.); and maintenance and operations (to perform the actual migration and data validation).

The design team has proposed a process to ensure the successful migration of Version 0 data into the ECS. The process is documented in a separate Data Migration Plan white paper (160-TP-002-001, Version 1). That white paper describes the process by which Version 0 data is transformed into a standard form that is recognizable by the Ingest Subsystem. The white paper assumes that the Ingest Subsystem is structured to facilitate ingest of data in a standard form.

As described in the following sections, the Ingest Subsystem is structured so that data is ingested into ECS by a standard mechanism once the data is preprocessed (e.g., data conversions and reformatting, metadata extraction, and metadata quality checking). Therefore, the primary task of Version 0 migration--data preprocessing--may be performed externally from the Ingest Subsystem. In addition, the Ingest Subsystem design provides standard ingest interfaces (as described in

previous subsections). Therefore, the selection of the data transfer mechanism (e.g., 8mm tape, network transfer) for a specific Version 0 data product may require additional copies of Ingest Subsystem hardware, but does not require new (i.e., undesigned) Ingest Subsystem hardware or software. The potential use of a data transfer media that is not already supported in Release A would need to be evaluated on a case-by-case basis.

Accordingly, the Version 0 migration effort may be planned as a separate activity from the Ingest Subsystem development. A proposal for development of a "Version 0 Migration Facility" was reported on at the Release B Interim Design Review (IDR).

### **3.2.2.4 Impact of Performance Requirements**

#### TRMM Expedited Data Ingest

The Ingest CSCI assigns different priorities to each external site (e.g., the SDPF). The capability to assign a different priority to each category of data within the site (e.g., production Level 0 data/ expedited (equivalent to quicklook) data) is available as an extension to the design, but is not currently implemented. Our analysis indicates that the rate of ingest of SDPF data (delivery of Level 0 data once per day; delivery of expedited data three times per day) and the relatively small data volumes (much less than one gigabyte) may be handled by the existing priority scheme.

#### TRMM I/O Throughput

Based upon Level 0 data volumes and loading estimated in Appendix A of the TRMM IRD ECS modeled the nominal data I/O throughput. Note: the non-Level 0 data flow into Data Server hardware, and therefore do not impact the Ingest HWCI. In addition, the following TRMM IRD requirement for TSDIS reprocessing imply the peak throughput:

TRMM4090 -- ...ECS shall also daily ingest an average of 2 days worth of reprocessed data from TSDIS. (GSFC interface)

Assuming that Level 0 data is retrieved for reprocessing at this same rate, the working storage component of the Ingest HWCI is sized for 2.5 days' data volume (one day's worth of Level 0 data ingest, one days' worth of Level 0 data retrieved from storage plus 25% contingency). The hardware sizing information is documented in section 5 of this document and in the DAAC-specific volumes for LaRC and MSFC.

#### EDOS Expedited Data Ingest

As with the SDPF expedited data, analysis indicates that the rate of ingest of EDOS expedited data (several times per day) and the relatively small data volumes (much less than one gigabyte) may be handled by the existing Ingest priority scheme in concert with ongoing ingest for EDOS and other data products.

#### EDOS I/O Throughput

Based upon Level 0 data volumes and loading specified in the February 1996 Technical Baseline ECS modeled the nominal data I/O throughput. Note: the Level 0 data flow into the Ingest HWCI.

The working storage component of the Ingest HWCI is sized for 2.5 days' data volume (one day's worth of Level 0 data ingest, one days' worth of Level 0 data retrieved from storage plus 25% contingency). The hardware sizing information is documented in section 5 of this document and in the DAAC-specific volumes for LaRC and GSFC.

### Landsat 7 (LPS) I/O Throughput

Based upon Level 0R data volumes and loading specified in the February 1996 Technical Baseline ECS modeled the nominal data I/O throughput. Note: The Level 0R data flow into the Ingest HWCI working storage component and are transferred to Data Server hardware for permanent storage.

The Ingest HWCI is sized for 1.25 days' data volume (one day input from LPS, 25% contingency). The Data Server HWCI is sized for an additional two days' data volume for retrieval of L0R data. The ICLHW hardware sizing information is documented in section 5 of this document and in the DAAC-specific volume for EDC. The Data Server hardware information is documented in the Data Server volume of this document.

### Other L0 I/O Throughput

Other Level 0 interfaces (ADEOS II/SeaWinds, FOO/COLOR, SAGE III, ALT RADAR, and ACRIMSAT) are comparable in data volume and loading with the TRMM Level 0 interface. They are proposed to be implemented similarly. Table 5.1-1 provides details.

## **3.2.3 Ingest Subsystem Design Paradigms**

The Ingest subsystem is characterized by the design paradigms used to facilitate the subsystem's construction. The Ingest subsystem uses four basic design paradigms--object factory creation of objects, multi-threading/prioritization of processes, distributed objects, and a client/server architecture. The following subsections discuss the Ingest subsystem's use of those concepts.

### **3.2.3.1 Object Factory**

The object factory approach is used in three areas of the Ingest subsystem. First, the Ingest Server object factory creates an Ingest Session object for each connection requested by an external data provider. Second, an Ingest Request Manager object factory creates an Ingest Request object for each request submitted by the external data provider. Third, an Ingest Granule Server object factory creates an Ingest Granule object for each granule specified in a request. Each Ingest Session, Ingest Request, and Ingest Granule object performs its actions asynchronously from other objects. The Ingest Session, Ingest Request, and Ingest Granule objects together perform the fundamental activities required to ingest data into ECS.

See section 4.5 of this volume for details on the object factory implementation.

### **3.2.3.2 Multi-threading/Prioritization**

DCE pthreads are used to implement multithreading for the Ingest Request and Ingest Granule objects. Prototypes have verified that several thousand pthreads may be generated per Unix process.

The Ingest subsystem implements request prioritization based on DCE pthread priority mechanisms. The pthread round-robin prioritization scheme was selected. The round-robin scheme provides time-sliced processor access for pthreads of equal priority. The highest-priority requests are processed to completion or until blocked, at which time lower-priority requests are processed.

The default priority for a request is obtained from an internal table, based on the external data provider. Operations personnel have the capability to change the priority of an ongoing request once it is received by the Ingest software.

See section 4.5 of this volume for details on the pthread and prioritization implementation. See Section 5.4.8, Volume 0, of this document for a general discussion of the DCE pthread concept.

### **3.2.3.3 Distributed Objects**

The Ingest Server, Request Manager, and Granule Server object factories are fully distributed. A given ECS site contains a single instance of the Ingest Server and Request Manager objects. A separate Granule Server is implemented on each processor on which data preprocessing occurs. Operator GUI services access the object factory objects using the DCE Name Service services of OODCE without regard to the processor on which GUI and object factory are installed.

### **3.2.3.4 Client/Server Architecture**

The Ingest subsystem contains a number of sequential client/server relationships. First, the external data provider (or the CSS Gateway that acts as interpreter for an external data provider) is the client to the Ingest Session object that services the data provider. Second, the Ingest Session object acts as a client to submit data provider requests to an Ingest Request server object. Third, each Request object acts as a client to submit data granule sub-requests to an Ingest Granule server object, which ingests the granule. Finally, the Ingest Granule object acts as a client to invoke Data Server subsystem servers that perform working storage allocation, data transfer, and data insertion.

## **3.2.4 Ingest Subsystem Use of Key Design Mechanisms**

ECS provides a number of key design mechanisms to facilitate implementation of the design paradigms described above. The key design mechanisms are documented in the CSS and MSS volumes of this document. The key design mechanisms represent software that is of general use across ECS subsystems as a common infrastructure. Use of the key design mechanisms ensures that standard implementations of common infrastructure components are used throughout ECS. The following subsections document the use of key design mechanisms by the Ingest subsystem.

### **3.2.4.1 Server Request Framework (SRF)**

The ECS Server Request Framework (SRF) provides a standard implementation of the object factory and pthread paradigms described above. Ingest subsystem object classes are derived from base SRF classes to implement the Ingest Granule Server/Ingest Granule object classes. The SRF also guarantees asynchronous message delivery between objects implemented by means of the SRF.

The Server Request Framework overlays the ECS Distributed Object Framework (DOF). The Server Request Framework is discussed in Section 5.4.8 and 6.1.5 of Volume 0 of this document. The Distributed Object Framework is discussed in Sections 5.4.8 and 6.1.2 of Volume 0.

### **3.2.4.2 End-to-end Request Tracking**

The Ingest Request object described above is a specialization of a general ECS request object--EcRequest. The EcRequest object generates a unique request ID that is used a) to track ingest requests from submission through archiving of data granules and b) provide a linkage to events

logged in the MSS event log. The request ID is displayed on Ingest subsystem status monitoring screens and stored in the Ingest History Log, which collects summary data on the ingest requests. The request ID is also attached to each event logged using the MSS event logging service. Together, the request ID allows operations personnel to track and analyze ingest requests at all stages of ingest processing.

#### **3.2.4.3 Universal References/Advertising**

Universal References (URs) are used to uniquely identify critical ECS entities. The Ingest subsystem uses Data Server subsystem service URs and data granule URs.

The Ingest subsystem uses Data Server service URs to locate available Data Server insert and resource management services. The URs are obtained by use of an Ingest subsystem tool invoked at initial system startup and re-invoked thereafter whenever the set of available URs changes. The tool obtains all available URs from the Advertising service, based on the full set of data types that may be inserted into the Data Server subsystem. The URs are stored in a local Sybase table. Note: if URs change, but the tool is not used, subsequent Ingest requests may fail.

The Ingest subsystem uses Data Server data granule URs to identify archived data after insertion into the Data Server subsystem. The URs may be returned to data providers by means of email. Data providers may use the URs to access data within the Data Server subsystem.

Universal References are discussed in Section 6.1.3 of Volume 0 of this document.

#### **3.2.4.4 Process Framework**

Unix processes created to invoke the Ingest subsystem components are managed by the MSS Process Framework. All processes that make use of process management and event logging services invoke the MSS Process Framework.

The Ingest subsystem overloads the startup and shutdown services of the Process Framework. The Ingest startup service allows the Ingest subsystem components to start up in an appropriate order. The Ingest Granule Server is created first, the Ingest Request Manager process is created second, and the Ingest Server is created last. The Ingest shutdown service allows the Ingest subsystem components to shut down gracefully on operations staff command. For the Ingest subsystem, the order of process shutdown is immaterial. Ingest subsystem recovery mechanisms, described below, allow subsequent restart without concern for the relative state of processes during shutdown.

The Process Framework is discussed in Sections 5.4.8 and 6.1.4 of Volume 0 of this document.

#### **3.2.4.5 Event Detection/Logging**

The Ingest subsystem makes full use of CSS event logging and browsing services. The classes of events reported and the use of CSS services are described in subsection 4.6.1.2 of this document.

Event, exception, error and fault handling (and their definitions) are described in Sections 5.4.8 and 6.4.5 of Volume 0 of this document.

### 3.2.4.6 Mode Management

The Ingest subsystem uses CSS Mode Management services to allow instantiation of the Ingest subsystem in operational, testing, and training modes. The CSS Process Framework passes the mode parameter to the Ingest subsystem at system startup. The Ingest subsystem subsequently uses the mode parameter as a suffix to data base names, DCE object names, and system directory names. Separate Sybase data bases and Unix system directories are implemented for each available mode. The mode parameter is also stored in selected Sybase tables to distinguish the mode in which the table entry was generated.

Operational, testing, and training mode instantiations of the Ingest subsystem software may individually run on any hardware on which the Ingest subsystem is installed. At ECS sites where an Ingest subsystem Hardware Configuration Item (HWCI) is implemented, primary and backup processors are available. The operational software typically runs on the primary, while testing and training instantiations run on the backup. Note: see the description of Ingest subsystem hardware/software failover in a subsequent subsection.

Mode management is discussed in Section 5.4.9 of Volume 0 of this document.

### 3.2.4.7 Performance and Accountability Reporting

The Ingest subsystem provides performance and accountability information to MSS by means of MSS performance and accountability reporting services. MSS provides agents to collect the provided performance and accountability information. MSS also provides SNMP agents to monitor configuration information related to Ingest subsystem hardware.

Table 3.2-2 lists the performance and accountability parameters collected and their frequency of collection. Performance and accountability reporting are discussed in Sections 6.4.6.3.2 and 6.4.6.3.3 of Volume 0 of this document.

**Table 3.2-2. Ingest Performance and Accountability Parameters (1 of 2)**

Parameter/Event	Description	Recommended Sample Frequency/ Event Frequency
Request Event	Per request: <ul style="list-style-type: none"><li>- Order ID</li><li>- External Data Provider</li><li>- Mission</li><li>- Total processing time</li><li>- Total data volume</li><li>- Total number of files</li><li>- Total number of granules</li><li>- Ingest type (automated, interactive, polling files, polling delivery record, media)</li><li>- Total Time to Transfer</li><li>- Total Time to Preprocess</li><li>- Total time to Archive</li><li>- Warm Start Flag</li><li>- Completion Status (successful or with errors)</li></ul>	Once per Ingest request

**Table 3.2-2. Ingest Performance and Accountability Parameters (2 of 2)**

Parameter/Event	Description	Recommended Sample Frequency/ Event Frequency
Request Granule Event	Per granule: <ul style="list-style-type: none"> <li>- Order ID</li> <li>- Data volume</li> <li>- Number of files</li> <li>- Time to transfer</li> <li>- Time to preprocess</li> <li>- Time to archive</li> <li>- Warm start Flag</li> <li>- Final Status (used to provide fault metrics)</li> </ul>	Once per request granule
External Data Provider Request Threshold	Maximum number of request allowed to be processed by a single data provided. A separate maximum exists for each data provider	Once per minute
External Data Provider Volume Maximum	Maximum volume of data allowed across all ongoing ingest requests submitted by an external data provider. A separate maximum volume exists for each external data provider	Once per minute
Data Volume Buffered	Total volume of data being throttled, by Ingest Request Processing, due to excessive system/external data provider volume loads	Once per minute
Data Requests Buffered	Total number of requests being throttled, by Ingest Request Processing, due to excessive system/external data provider request load	Once per minute
Daily Count of Data Quality Faults	Running count of data quality faults. Initialized to zero at Ingest process initialization. Reset to zero once a day by Ingest software. Data quality errors include metadata missing or invalid values, missing required files or DAN file validation failures. Note: detail error status values are reported in Request granule event described above.	Once every fifteen minutes
Daily Count of Ingest Errors	Running count of Ingest errors. Initialized to zero at Ingest process initialization. Reset to zero once a day by Ingest software. Ingest errors include all failures, excluding data quality faults, which cause a granule in a request to fail to be archived (e.g., data base access failures, external communication errors, system failures, request rejections for security failures)	Once every five minutes
Daily Count of Request Processed (successfully or unsuccessfully)	Running count of all Ingest requests. Initialized to zero at Ingest process initialization. Reset to zero once a day by Ingest software.	Once every fifteen minutes

### **3.2.4.8 User Profile**

The Ingest subsystem uses the MSS User Profile to determine the authorization of a specified user to ingest data into ECS. A separate entry is included in the User Profile for each authorized science user as well as for each institutional data provider (e.g., TSDIS and the Landsat-7 Processing System).

The User Profile also provides default email addresses for each science user/institutional data provider. The email address is used if a data provider has requested notification of ingest status or provision of URs by means of email.

The User Profile is discussed in Section 5.4.9 of Volume 0 of this document.

### **3.2.5 Ingest Subsystem Key Design Features**

The Ingest subsystem provides a number of additional features that optimize the reliability, availability, maintainability, and security of the subsystem software and hardware. Together, the features provide the robustness and flexibility required to support the Ingest subsystem throughout its project lifetime.

#### **3.2.5.1 Ingest Subsystem Failover**

As described above, certain ECS sites have highly-reliable Ingest subsystem hardware, containing primary and backup processors. The processors have cross-strapped connections to working storage devices and Sybase data base disks. Working storage devices are RAID-5, while disks used for Sybase transaction log storage are RAID-1 (mirrored). The Ingest subsystem provides a switchover capability that transfers execution of the operational software from the primary to the backup processor in the event of a primary failure. In that eventuality, testing and training instantiations on the backup processor are shut down by the Process Framework.

An MSS agent is implemented to detect a system failure of the primary system hardware. In addition, the an MSS agent detects critical software failures.

Upon operations staff command the MSS Process Framework restarts the backup processor with an identical Internet address to that of the primary processor. Upon Process Framework software startup, the Ingest subsystem uses its recovery capabilities to restart Ingest subsystem software from their checkpointed states. Thereafter, the backup processor instantiation of the Ingest subsystem is identical to the previous instantiation on the primary processor.

#### **3.2.5.2 Tunable Parameters**

The Ingest subsystem provides the capability to tune certain parameters. Parameters are stored in the Ingest Configuration File or passed as a calling parameter to a program at system startup or are stored in an Ingest data base table. Table 3.2-3 lists the tunable parameters and their source.

Parameters stored in the Configuration File are loaded into an MSS Management Information Block (MIB). Parameters in the MIB can be changed during run time by means of an MSS GUI.

Calling parameters are only changeable at system startup by means of operator input to the Process Framework.

Parameters stored in an Ingest data base table are changed via the Ingest Operator Tool GUI screens.

### 3.2.5.3 Recovery

The Ingest subsystem is designed to allow automatic recovery from system failures in the Ingest subsystem hardware or software CIs. The Sybase data base is the primary mechanism facilitating recovery. The Ingest Server, Ingest Session, Request Manager, Ingest Request, Granule Server, and Ingest Granule objects each checkpoint state information in Sybase. In the event of a warm system restart after a processor or software component failure, each checkpointed object reinitializes itself based on the checkpointed information. Once an ingest request is completed, summary information is transferred to the Ingest History Log (also maintained in Sybase) using Sybase triggers, and the original checkpoint information is deleted.

Request level checkpointing information is entered when an ingest request is received. Granule level checkpointing information is entered as each granule referenced in a request is transferred, preprocessed, and inserted into the Data Server subsystem. File level checkpointing information is entered after each file in a granule is transferred. Completed files, granules, or requests are not restarted during system restart.

**Table 3.2-3. Ingest Tunable Parameters**

Parameter / Parameter Category	Description	Change Control	Source
Polling Ingest Timer	The time period which indicates how often the Polling Ingest Session should check for the existence of ingest files for ingest processing. A separate polling timer exists for each external data provided using the Polling Ingest Mechanism	Ingest GUI	Ingest Data Base Table
External Data Provider Priority	Priority associated with ingest requests from the external data provider. A separate priority exists for each external data provider	Ingest GUI	Ingest Data Base Table
External Data Provider Request Maximum	Maximum number of requests allowed to be processed at one time by the external data provider. A separate maximum exists for each external data provider.	HP Open View	Ingest Configuration File
External Data Provider Volume Maximum	Maximum volume of data allowed across all ongoing ingest requests submitted by an external data provider. A separate maximum volume exists for each external data provider.	HP Open View	Ingest Configuration File
Communication Retry Count	Number of retries to perform when a communication failure is encountered with the external data provider.	Ingest GUI	Ingest Data Base Table
MonitorTimeFor CompletedRequest	Time (in minutes) a completed request will remain on the operators monitor screen	Ingest GUI	Ingest Data Base Table
WarmStartTimeLimit	Time (in days) failed requests can be warm started from checkpointed state (checkpointed data exceeding this time limit will be automatically deleted from the Ingest checkpoint tables)	Ingest GUI	Ingest Data Base Table

**Table 3.2-3. Ingest Tunable Parameters**

Mode Parameter	Determines in which mode (operational, test, training) to initiate an Ingest process. Supplied by operations personnel to the MSS Process Framework at system startup	Process Framework	Calling Parameter
Startup Mode	Determines if Ingest software is warm or cold started	Process Framework	Calling Parameter

Both warm restart and cold restart capabilities are supported by the MSS Process Framework. A parameter is entered at system startup to indicate which restart capability to invoke. If cold restart is selected, then checkpointed information is deleted, and ingest processing continues with new ingest requests. If warm restart is selected, then Ingest subsystem components are reinitialized as described above.

Specific details on error detection and reporting and on fault tolerance and error recovery are provided in sections 4.6.1.2 and 4.6.1.3, respectively, of this document. Table C-1 in Appendix C shows the major fault categories and their recoverability options. Fault categories are marked as "warm restart recoverable", "immediate retry recoverable", and "unrecoverable".

For warm restart recoverable faults, the Ingest subsystem attempts to restart ingest processing on each checkpointed granule. If the restart fails, the error is reported and the recovery attempt is aborted for the given granule.

Retries are performed when the Ingest software sends a Data Delivery Notice (DDN) to an external data provider. The number of retries and the wait time between retries is stored in the Configuration File.

Unrecoverable faults include situations where a) the maximum retry limit has been exceeded; b) system volume and request thresholds have been exceeded; and c) a COTS hardware or software component has failed and cannot be restarted (e.g., after a hard disk error or Sybase failure). Where backup information is retained (e.g., Sybase transaction logs), data may be manually reloaded and recovery attempted after system restart.

Fault tolerance and recovery are discussed in Section 6.4.5.2.4 of Volume 0 of this document.

### **3.2.5.4 Security**

The Ingest subsystem makes use of standard ECS security services to prevent unauthorized ingest of data. The Ingest subsystem is protected in three ways--by means of user authorizations, by use of kftp, and by use of CSS wrappers to Kerberos security services.

Institutional data providers, including authorized science users, may ingest data into ECS. Authorization information for data providers is stored in the MSS User Profile. The Ingest subsystem verifies the authorization of a data provider when a) preparing to ingest data from supplied media and b) receiving ingest information from the Interactive Ingest GUI. In addition, the CSS Ingest Gateway authenticates access by automated network ingest data providers (e.g., TSDIS, SDPF, Landsat-7 LPS).

Per NASA security guidelines Kerborized ftp (kftp) is the standard file transfer mechanism for external data providers seeking to transfer data into ECS. Kftp accesses Kerberos security services to request file transfer resources using an encrypted key mechanism. Note: several external data providers have sought and received a waiver on the use of kftp and are using ftp instead. The username/password information required for ftp is not encrypted and is therefore less secure than for kftp use.

To enhance security each data provider authorized to transfer data into ECS is assigned a directory location. The directory location is stored in the User Profile. The directory is protected by means of Unix file permissions.

In addition, the DCE services which underly interprocess communications within ECS use CSS security services (which use Kerberos security services) to authenticate application use of ECS services. Unauthorized external applications attempting to access ECS services are rejected.

Finally, the EDOS interface to the Ingest subsystem is implemented behind a firewall that prevents access of EDOS systems by users logged in through the Ingest subsystem processors.

The ECS security architecture is discussed in Section 6.2 of Volume 0 of this document.

### **3.2.5.5 Parallel Rel A/B Operations**

Ingest subsystem mechanisms for transitioning from Release A to Release B are described in a separate technical paper, 420-TP-010-001.

### **3.2.5.6 Portability**

Ingest subsystem components are implemented to facilitate portability to multiple Unix platforms. Components are implemented using Posix 1003.1a, which provides standard implementations of Unix system services. Currently, software is developed in a Sun Solaris environment and integrated on the target SGI Irix environment.

### **3.2.5.7 Data Backup Policy**

Since all Ingest subsystem system-level data is stored in the Sybase data base, backup may be accomplished by means of standard Sybase backup utilities and load tools. This includes all information stored in the Ingest History Log, which summarizes Ingest activity.

### **3.2.5.8 Application Programming Interfaces**

Application Program Interfaces (APIs) are provided to selected Ingest subsystem services--create session, create request, cancel request, hold request, and resume request. In addition, all Ingest subsystem system-level data is stored in the Sybase data base. DAACs may develop DAAC-unique interfaces using the APIs and/or Sybase interfaces to a) implement new polling interface mechanisms; b) implement new GUI interfaces, including new reports; or c) other DAAC-specific applications.

### **3.2.5.9 Summary Statistics Reports**

The summary statistics reports available from the Ingest subsystem are discussed in section 4.6.3 of this document.

## 4. INGST - Ingest CSCI

---

### 4.1 CSCI Overview

The Ingest CSCI is responsible for the receipt of data arriving at a site and the physical placement of data into the site's storage hierarchy. A provider site within EOSDIS will normally need to ingest a wide variety of data types to support the services it wishes to offer. These data may be delivered through different interfaces (network file transfer, hard media, hard copy, etc.), with varying management approaches to these interfaces. This interface heterogeneity and the need to support extendability and new data/interfaces as algorithms and provider functionality changes, lead to a design in which the ingest functionality is isolated from other subsystems within the segment design.

Although each instance of the Ingest CSCI has to deal with the characteristics of the specific external interface it is managing, the general functionality is similar in each case.

- Ingest processing is either event-driven or timer-driven. For automated network ingest (e.g., SDPF and Landsat-7 LPS), data centers send Data Availability Notices to the DAACs to indicate the availability of data. For hard media ingest, the "data availability notice" is entered by DAAC operations staff at a GUI interface. Similarly, for interactive network ingest under science user control, the "data availability notice" is entered by the science user at a GUI interface. For timer-driven ingest (e.g., NESDIS and EDOS), on the other hand, data centers transfer data to an agreed-upon network location and ECS ingest clients periodically check ("poll") for the existence of new data.
- Depending on the interface, data may be transferred by either a data "get" or a data "put". A data get is performed by the Ingest CSCI under Ingest CSCI control. A data put is performed by another data center under that data center's control.
- The Ingest CSCI performs transmission checks relevant to the transfer mechanism (e.g. data quality, data redundancy, missing files, etc.) and notifies the data source of success or failure. Failure results in a request to resend or in notification of the operations staff. The DAAC operations staff monitors the status of active ingest processing.
- The Ingest CSCI extracts sufficient metadata to allow the data to be retrieved at a later time from the Data Server. The metadata information is contained within the data file, within a separate metadata file in standard format associated with the data, or within the information provided to request ingest. The form in which metadata is provided is determined by the Interface Control Document (ICD) defined for the specific interface. Some portion of the metadata is checked for quality (e.g., all required metadata parameters available, parameters within a range of values, etc.). Additional metadata, such as the time of ingest, is determined by the Ingest CSCI.
- When the collection of data is complete (i.e., all referenced data are available), the Ingest CSCI requests insertion of the data into an appropriate Data Server.
- The Ingest CSCI records the successful or unsuccessful transfer of data into the site in an ingest history log. The ingest history log stores summary information for each ingest

request accepted by ECS. Detailed error and status information is stored in an event log. Entries in the event log and in ingest history log are related by means of a date/time stamp and by means of the ingest request identifier. The DAAC operations staff and System Management Center (SMC) staff may interrogate the ingest history log and the event log. In addition, the Ingest CSCI returns the completion status for the ingest data transfer to the data ingest requester.

## 4.2 CSCI Context

The context diagram for the Ingest CSCI is identical to that of the Ingest Subsystem, since the Ingest CSCI is the only CSCI for the subsystem (see Figure 3.1-1). Table 4.2-1 shows the CSCI service interfaces provided to entities external to the CSCI.

**Table 4.2-1. Ingest CSCI Service Interfaces**

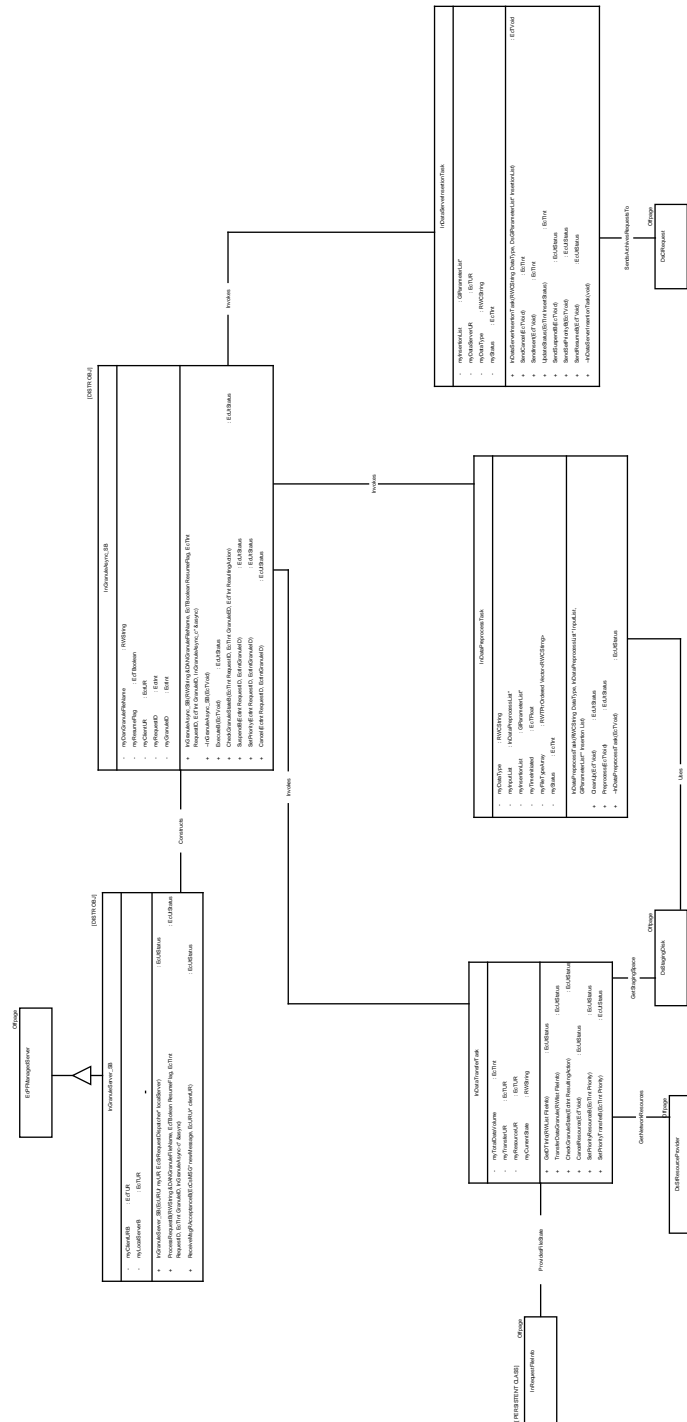
Interface	Input Data	Output Data	Description
Ingest Server. Create Session	External Data Provider	Ingest_Status	Sets up an ingest request session connection
Ingest Session. Receive Message	Data Availability Notice Data Delivery Ack	Data Availability Ack Data Delivery Notice	Provides the application-level protocol for automated network data transfer
Media Ingest Session. Receive Message	Hard Media Ingest Request	Ingest_Status	Provides authorized operations staff the means to enter, by means of GUI input, information required to ingest hard media
Network Ingest Session. Receive Message	Network Ingest Request	Ingest_Status	Provides authorized science users the means to enter, by means of GUI input, information required to ingest data by network data transfer
Status Monitor. Receive Request	User_Identifier	Ingest Request (list)	Provides the status of a) all or selected ongoing ingest requests (authorized operations staff) or b) ongoing requests for a given user (specific user)
Log Monitor. Receive Message	Ingest Log Request	Ingest Log	Provides the status of all or selected completed ingest requests to authorized operations staff

## 4.3 Ingest CSCI Object Model

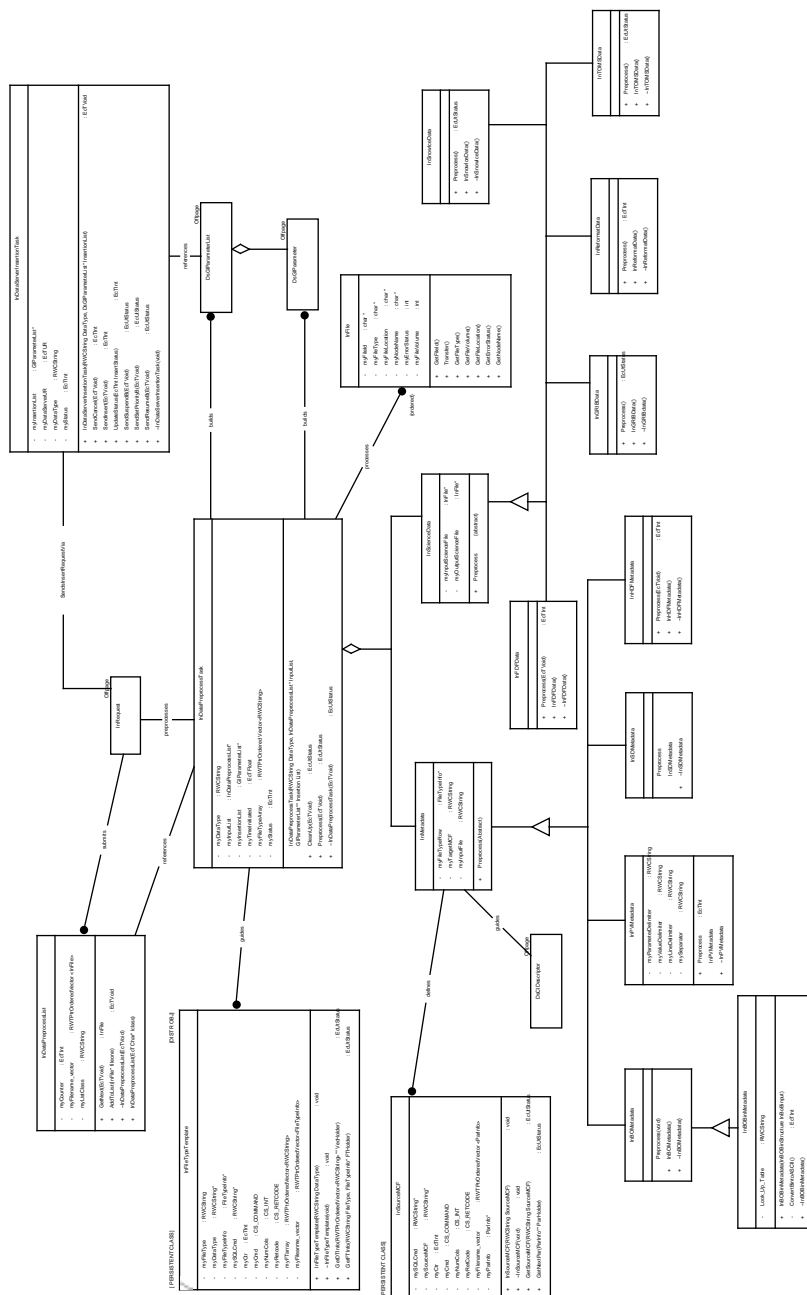
Figures 4.3-1 through 4.3-5 show the object classes that model the Ingest Subsystem. Subsequent paragraphs describe each of the object classes in terms of their parent class, purpose and description, and their critical attributes, operations, and associations. With regard to commercial off-the-shelf software (COTS), the Ingest CSCI uses:

- Rogue wave class library services pervasively throughout the design
- Sybase for relational data base services; and
- A Netscape Commerce http WEB server

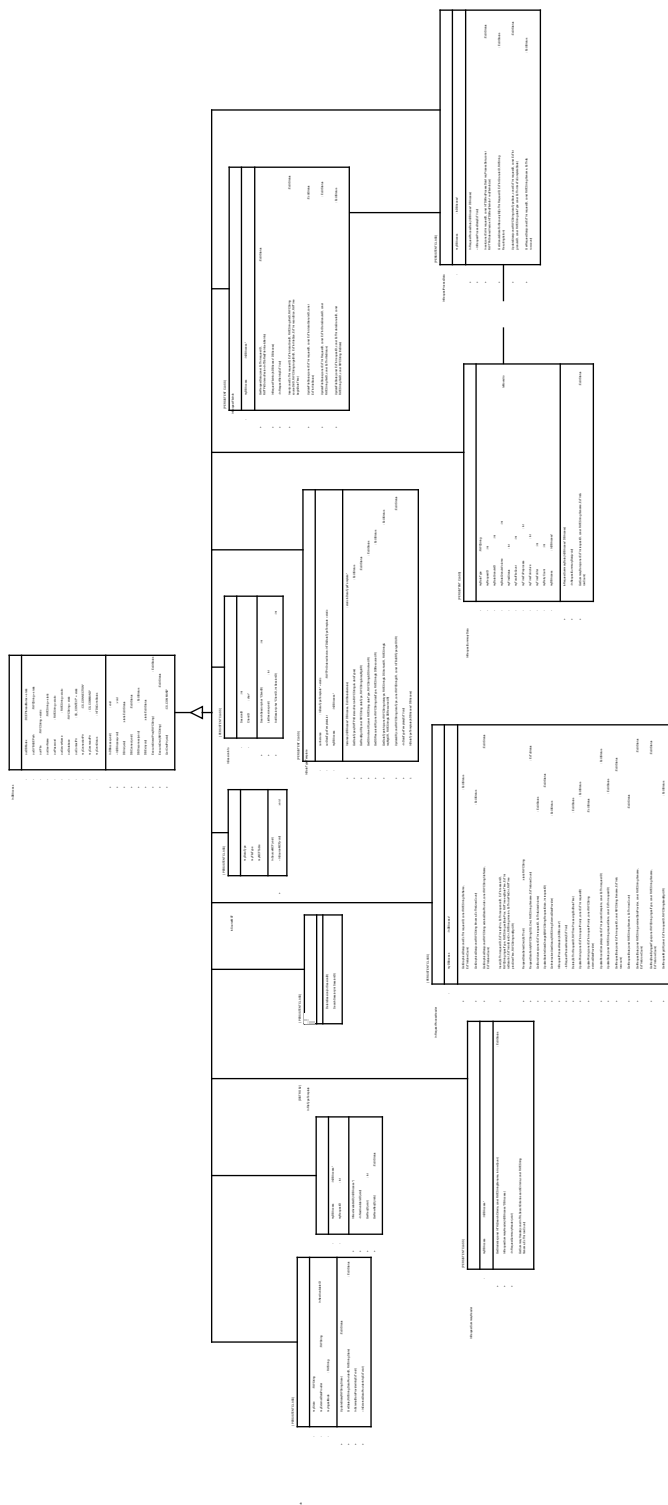




**Figure 4.3-2. In\_Ingest\_Granule Processing Object Model Diagram**



**Figure 4.3-3. In\_Ingest\_PreProcessing Object Model Diagram**



**Figure 4.3-4. In\_Ingest\_Database\_Object Model Diagram**



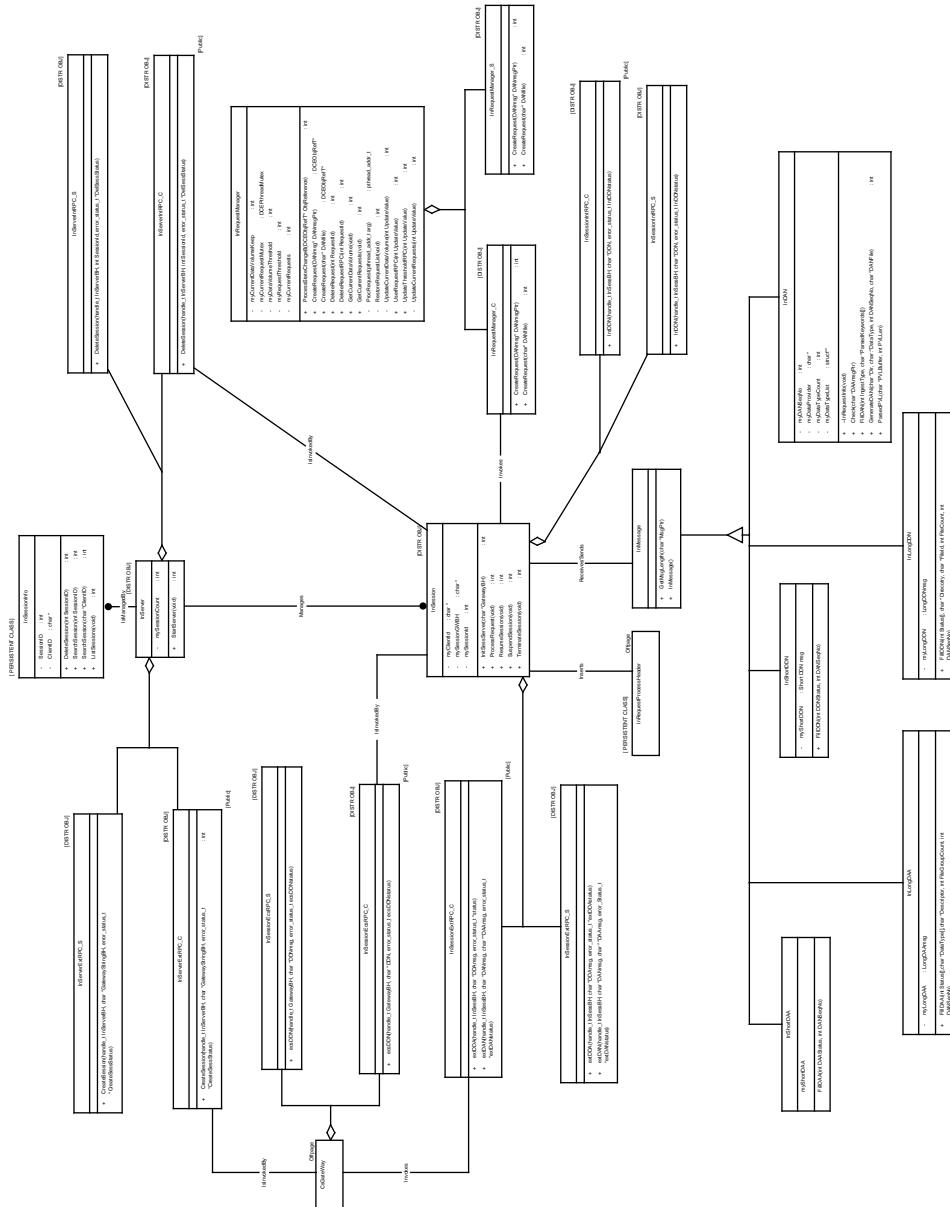


Figure 4.3-6. In\_Ingest\_Session\_Manager Object Model Diagram

### 4.3.1 CsGateWay Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

The Gateway translates TCP/IP socket call to the corresponding RPC function.

#### Attributes:

None

#### Operations:

None

#### Associations:

The CsGateWay class has associations with the following classes:

Class: InSessionExtRPC\_C Invokes - The Gateway object interfaces with the InSessionExtRPC object to deliver the DAN and DDA data messages received from the external Client to Ingest Session.

Class: InServerExtRPC\_C IsInvokedBy - The Gateway object interfaces with the InServerExtRPC to initiate a new Ingest Session through the Ingest Server.

### 4.3.2 DsCIDDescriptor Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

This class is a Data Server Subsystem class and therefore its attributes and operations are defined in the Data Server Subsystem documentation. The DsCIDDescriptor class provides the Preprocessing CSC services to access targetMCFs and validate metadata files.

#### Attributes:

None

***Figure 4.3-4. In\_Ingest\_Database\_Object Model Diagram***

**Operations:**

None

**Associations:**

The DsCIDescriptor class has associations with the following classes:

Class: InMetadata guides - The DsCIDescriptor class guides the InMetadata class by providing services to access target MCFs and validate metadata files.

**4.3.3 DsCIRequest Class**

Parent Class:Not Applicable

**Attributes:**

None

**Operations:**

None

**Associations:**

The DsCIRequest class has associations with the following classes:

Class: InDataServerInsertionTask SendsArchivesRequestsTo

**4.3.4 DsGIPParameter Class**

Parent Class:Not Applicable

**Attributes:**

None

**Operations:**

None

**Associations:**

The DsGIPParameter class has associations with the following classes:

Class: InDataPreprocessTask builds

DsGIPParameterList (Aggregation)

#### **4.3.5 DsGIPParameterList Class**

Parent Class:Not Applicable

**Attributes:**

None

**Operations:**

None

**Associations:**

The DsGIPParameterList class has associations with the following classes:

Class: InDataPreprocessTask builds

Class: InDataServerInsertionTask references

#### **4.3.6 DsStResourceProvider Class**

Parent Class:Not Applicable

**Attributes:**

None

***Figure 4.3-5. In\_Ingest\_Session Object Model Diagram***

**Operations:**

None

**Associations:**

The DsStResourceProvider class has associations with the following classes:

Class: InDataTransferTask GetNetworkResources

Class: InMediaIngest uses

**4.3.7 DsStagingDisk Class**

Parent Class:Not Applicable

**Attributes:**

None

**Operations:**

None

**Associations:**

The DsStagingDisk class has associations with the following classes:

Class: InDataTransferTask GetStagingSpace

Class: InDataPreprocessTask Uses

**4.3.8 EcPFManagedServer Class**

Parent Class:Not Applicable

**Attributes:**

None

**Operations:**

None

**Associations:**

The EcPFManagedServer class has associations with the following classes:

Class: InPollingIngestSession

**4.3.9 InBOBinMetadata Class**

Parent Class:InBOMetadata

Public:No

Distributed Object:No

Purpose and Description:

This class provides services to preprocess byte ordered binary data.

**Attributes:**

**Look\_Up\_Table** - This attribute defines the appropriate look-up table for conversion from binary to ASCII.

Data Type:RWCString

Privilege:Private

Default Value:

**Operations:**

**ConvertBintoASCII** - This operation converts binary data to ASCII values through the use of a data/file type specific look-up table.

Arguments:

Return Type:EcTInt

Privilege:Private

**InBOBinMetadata** - This is the object class constructor

Arguments:InBOBinStructure InBoBinput

Return Type:Void

Privilege:Public

**~InBOBinMetadata** - This is the object class destructor

Arguments:

Return Type:Void

Privilege:Public

#### **Associations:**

The InBOBinMetadata class has associations with the following classes:

None

#### **4.3.10 InBOMetadata Class**

Parent Class:InMetadata

Public:No

Distributed Object:No

Purpose and Description:

This class provides services to preprocess byte ordered data.

#### **Attributes:**

All Attributes inherited from parent class

#### **Operations:**

**InBOMetadata** - This is the constructor.

Arguments:

Return Type:Void

Privilege:Public

**Preprocess** - Performs the data preprocessing (e.g., mandatory keywords check, required files check)

Arguments:void

**~InBOMetadata** - This is the destructor.

Arguments:

Return Type:Void

Privilege:Public

## **Associations:**

The InBOMetadata class has associations with the following classes:

None

### **4.3.11 InDAN Class**

Parent Class:InMessage

Public:No

Distributed Object:No

Purpose and Description:

This is the DAN (Data Availability Notice) is received from the external Client. The object class contains services to access information in the DAN.

## **Attributes:**

**myDANSeqNo** - The identifier of the DAN data message.

Data Type:int

Privilege:Private

Default Value:

**myDataProvider** - Indicates who provides the data for ingest.

Data Type:char \*

Privilege:Private

Default Value:

**myDataTypeCount** - Indicates the total number of data types in the DAN.

Data Type:int

Privilege:Private

Default Value:

**myDataTypeList** - This is a list that contains information about the file (e.g., file name, size, location).

Data Type:struct\*\*

Privilege:Private

Default Value:

## **Operations:**

**Check** - Verifies the integrity of the DAN components.

Arguments:char \*DAAMsgPtr

Return Type:Void

Privilege:Public

### **CreateDANGranulefiles**

Arguments:RWString &DANFile

**FillDAN** - Fills the DAN information into the class data memory after the DAN information is parsed.

Arguments:int IngestType, char \*ParsedKeywords[]

Return Type:Void

Privilege:Public

**GenerateDAN** - Generates a DAN file with file information retrieved from the given directory location.

Arguments:char \*Dir, char :\*DataType, int DANSeqNo, char \*DANFile

Return Type:int

Privilege:Public

**ParsedPVL** - Extracts information from the DAN and puts the information into a data memory.

Arguments:char \*PVLBuffer, int PVLLen

Return Type:Void

Privilege:Public

## **Associations:**

The InDAN class has associations with the following classes:

Class: InRequest Defines

### **4.3.12 InDBAccess Class** - This class provides services to access the Ingest data base

Parent Class:Not Applicable

## Attributes:

**myCommandPtr** - Database command pointer.

Data Type:CS\_COMMAND\*

Privilege:Private

Default Value:

**myConnStatus** - Connection status.

Data Type:InTDbConnStatus

Privilege:Private

Default Value:

**myConnectionPtr** - Database connection pointer.

Data Type:CS\_CONNECTION\*

Privilege:Private

Default Value:

**ourContextPtr** - Database context pointer.

Data Type:CS\_CONTEXT\*

Privilege:Private

Default Value:static

**ourDBMutex** - Database mutex.

Data Type:DCEPthreadMutex

Privilege:Private

Default Value:static

**ourDatabase** - Ingest database name.

Data Type:RWCString

Privilege:Private

Default Value:static

**ourIFile** - SYBASE interface file.

Data Type:RWCString

Privilege:Private

Default Value:static

**ourPassword** - SYBASE password.

Data Type:RWCString

Privilege:Private

Default Value:static

**ourSYBASEPath** - SYBASE path.

Data Type:RWCString

Privilege:Private

Default Value:statis

**ourServerName** - SYBASE servername.

Data Type:RWCString

Privilege:Private

Default Value:static

**ourUserName** - SYBASE username.

Data Type:RWCString

Privilege:Private

Default Value:static

## **Operations:**

**DBConnect** - Connect to the Ingest database.

Arguments:void

Return Type:EcUtStatus

Privilege:Public

**DBDisconnect** - Disconnect from the Ingest database.

Arguments:void

Return Type:EcUtStatus

Privilege:Public

**DBExit** - Exits all connections to the Ingest database.

Arguments:void

Return Type:static EcUtStatus

Privilege:Public

**DBInit** - Initialize STYBASE Client-Library.

Arguments:void

Return Type:statis EcUtStatus

Privilege:Public

**ExecuteCmd** - Execute SQL statement.

Arguments:RWCString

Return Type:EcUtStatus

Privilege:Public

**ExecuteStoredProc** - Execute an SQL command to a call stored procedure which has no output parameters.

Arguments:RWCString

Return Type:EcUtStatus

Privilege:Public

**GetCmdPtr** - Returns pointer to the command structure.

Arguments:void

Return Type:CS\_COMMAND\*

Privilege:Public

**InDBAccess** - Default constructor.

Arguments:void

Return Type:void

Privilege:Public

**~InDBAccess** - Destructor.

Arguments:void

Return Type:void

Privilege:Public

#### **Associations:**

The InDBAccess class has associations with the following classes:

None

#### **4.3.13 InDataPreprocessList Class**

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

The purpose of this class is to retain lists (of files). This class provides services to add files to an existing list and retrieve files from an existing list.

## Attributes:

**myCounter** - This attribute is a local counter for the class.

Data Type:EcTInt

Privilege:Private

Default Value:

**myFilename\_vector** - This attribute specifies the local Rogue Wave ordered vector.

Data Type:RWTPtrOrderedVector <InFile>

Privilege:Private

Default Value:

**myListClass** - This attribute identifies whether the list is an input list received from the Request Processing CSC or is a list containing files to be inserted into the Data Server Subsystem.

Data Type:RWCString

Privilege:Private

Default Value:

## Operations:

**AddToList** - This service provides the ability to add a file to an existing list.

Arguments:InFile\* fileone

Return Type:EcTVoid

Privilege:Public

**GetNext** - This service provides the ability to retrieve the next file in the list.

Arguments:EcTVoid

Return Type:InFile

Privilege:Public

**InDataPreprocessList** - This is the constructor.

Arguments:EcTChar\* lclass

Return Type:Void

Privilege:Public

**~InDataPreprocessList** - This is the destructor.

Arguments:EcTVoid

Return Type:Void

Privilege:Public

## Associations:

The InDataPreprocessList class has associations with the following classes:

Class: InDataPreprocessTask references

Class: InRequest submits - The InRequest class submits InDataPreprocessList in order to identify the files which need to be preprocessed.

### 4.3.14 InDataPreprocessTask Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Persistent Class:

Purpose and Description:

The main purpose of this class is to initiate and monitor required data preprocessing before insertion into the data server subsystem. The InRequest Class instantiates this class for each separate preprocessing task. The InRequest class will supply an object ID for the input file list (which contains the files associated with the preprocessing task). The InDataPreprocessTask Class instantiates the InDataType Class. This class is responsible for the control and reporting of its assigned preprocessing as directed by the InRequest Class. It is also responsible for reporting the state of a particular preprocessing task whenever the state changes. This object class also provides services to cancel, suspend, and resume preprocessing tasks.

## Attributes:

**myDataType** - The attribute specifies the data type (e.g. metadata, science) for the Data Preprocess Task object.

Data Type:RWCString

Privilege:Private

Default Value:

**myFileTypeArray** - This attribute specifies the structure which stores constituent file type information for a specific data type.

Data Type:RWTPtrOrdered Vector<RWCStirng>

Privilege:Private

Default Value:

**myInputList** - This attribute references the input list which contains the files to be preprocessed for the data preprocess task object.

Data Type:InDataPreprocessList\*

Privilege:Private

Default Value:

**myInsertionList** - This attribute references the list which contains the processed files to be inserted into the Data Server Subsystem.

Data Type:GlParameterList\*

Privilege:Private

Default Value:

**myStatus** - This attribute specifies the status.

Data Type:EcTInt

Privilege:Private

Default Value:

**myTimeInitiated** - This attribute defines the initiation time of the Preprocessing Task.

Data Type:EcTFloat

Privilege:Private

Default Value:

## Operations:

**CleanUp** - This service will remove all files associated with checkpoint working storage.

Arguments:EcTVoid

Return Type:EcUtStatus

Privilege:Public

### InDataPreprocessTask

Arguments:RWCString DataType, InDataPreprocessList\* InputList, GlParameterList\*\* Insertion List

**Preprocess** - This service provides the management of preprocessing of data with the Preprocessing CSC. The service verifies the data type and its constituent file types; invokes the correct preprocessing classes to extract metadata and reformat/convert science data; and packages the modified data for subsequent insertion into the Data Server Subsystem.

Arguments:EcTVoid

Return Type:EcUtStatus

Privilege:Public

**~InDataPreprocessTask** - This is the destructor.

Arguments:EcTVoid

Return Type:EcUtStatus

Privilege:Public

#### **Associations:**

The InDataPreprocessTask class has associations with the following classes:

Class: InGranuleAsync\_SB Invokes

Class: DsStagingDisk Uses

Class: DsGIPParameter builds

Class: DsGIPParameterList builds

Class: InFileTypeTemplate guides

Class: InRequest preprocesses

Class: InFile processes

Class: InDataPreprocessList references

### **4.3.15 InDataServerInsertionTask Class**

Parent Class:Not Applicable

Public:No

Distributed Object:No

Persistent Class:

Purpose and Description:

Responsible for the sending of request to the appropriate Data Server based on the data type identifier. The object class has knowledge of the interface protocol with the Data Server and interfaces with the Advertising Service of the Interoperability Subsystem to determine the appropriate Data Server.

#### **Attributes:**

**myDataServerUR** - This is the Universal Reference of the Data Server to which the data is to be inserted.

Data Type:EcTUR

Privilege:Private

Default Value:

**myDataType** - This is the Universal Reference of the Data Server to which the data is to be inserted.

Data Type:RWCString

Privilege:Private

Default Value:

**myInsertionList** - List of files to be inserted into Data Server.

Data Type:GIParameterList\*

Privilege:Private

Default Value:

**myStatus** - Status of data insertion.

Data Type:EcTInt

Privilege:Private

Default Value:

## Operations:

**InDataServerInsertionTask** - This is the object class constructor.

Arguments:RWCString DataType, DsGIParameterList\* InsertionList

Return Type:EcTVoid

Privilege:Public

**SendCancel** - Sends insert cancellation request to the Data Server.

Arguments:EcTVoid

Return Type:EcTInt

Privilege:Public

**SendInsert** - Sends insert request to the Data Server.

Arguments:EcTVoid

Return Type:EcTInt

Privilege:Public

**SendResumeB** - Sends resume request to the Data Server.

Arguments:EcTVoid

Return Type:EcUtStatus

Privilege:Public

**SendSetPriorityB** - Sends set priority request to the Data Server.

Arguments:EcTVoid

Return Type:EcUtStatus

Privilege:Public

**SendSuspendB** - Sends suspend request to the Data Server.

Arguments:EcTVoid

Return Type:EcUtStatus

Privilege:Public

**UpdateStatus** - Receives notification from Data Server when an insert request has completed. Updates data base tables with completion information.

Arguments:EcTInt InsertStatus

Return Type:EcTInt

Privilege:Public

**~InDataServerInsertionTask** - This is the object class destructor.

Arguments:void

Return Type:Void

Privilege:Public

#### **Associations:**

The InDataServerInsertionTask class has associations with the following classes:

Class: InGranuleAsync\_SB Invokes

Class: DsCIRequest SendsArchivesRequestsTo

Class: InRequest SendsInsertRequestVia - The InRequest class SendsInsertRequestVia the InDataServerInsertionTask class to initiate the process necessary to insert data into the Data Server Subsystem.

Class: DsGIPparameterList references

#### **4.3.16 InDataTransferTask Class**

Parent Class:Not Applicable

Public:No

Distributed Object:No

Persistent Class:

Purpose and Description:

It is responsible for coordinating the data transfer and the population of the InTransferredData object class. In addition, the object class is responsible for attempting the data transfer retries when data transmission failed.

## Attributes:

**myCurrentState** - Contains current state information. In particular set to AllocatingResource, TransferringFiles, or Processing. Prior to setting to AllocatingResource or TransferringFiles, the corresponding UR has been set so that calls to Cancel or SetPriority can be processed.

Data Type:RWString

Privilege:

Default Value:

**myResourceUR** - Universal Reference of Resource Allocation Process. Used to Cancel or set priority of resource allocation.

Data Type:EcTUR

Privilege:Private

Default Value:

**myTotalDataVolume** - The total data volume of ingest files transmitted.

Data Type:EcTInt

Privilege:Private

Default Value:

**myTransferUR** - Universal Reference of File Transfer Processor

Data Type:EcTUR

Privilege:Private

Default Value:

## Operations:

**CancelResource** - Sends cancel request to Data Server resource allocation software using myResourceUR

Arguments:EcTVoid

Return Type:EcUtStatus

Privilege:Public

**CheckGranuleState** - Checks granule change state. If set to suspended or cancelled, this routine will update the Ingest data base plus set resulting action flag to indicate cancel request.

Arguments:EcTInt ResultingAction

Return Type:EcUtStatus

Privilege:Public

**GetDTInfo** - Returns the Data Type information of the files to be transferred

Arguments:RWList FileInfo

Return Type:EcUtStatus

Privilege:Public

**SetPriorityResourceB** - Sends set priority command to Data Server Resource software using myResourceUR.

Arguments:EcTInt Priority

Return Type:EcUtStatus

Privilege:Public

**SetPriorityTransferB** - Sends set priority command to Data Server Transfer software using myTransferUR

Arguments:EcTInt Priority

Return Type:EcUtStatus

Privilege:Public

**TransferDataGranule** - Handles the transfer of all files in a granule. Update ingest data base with transfer status of each file.

Arguments:RWlist FileInfo

Return Type:EcUtStatus

Privilege:Public

### **Associations:**

The InDataTransferTask class has associations with the following classes:

Class: DsStResourceProvider GetNetworkResources

Class: DsStagingDisk GetStagingSpace

Class: InGranuleAsync\_SB Invokes

Class: InRequestFileInfo ProvidesFileState

### 4.3.17 InDataTypeTemplate Class

Parent Class:InDBAccess

Public:No

Distributed Object:No

Persistent Class:True

Purpose and Description:

This class contains information that categorizes each ingest data type. The services provided enable the InDataType Class to determine what the required file types are for a given data type. The class will enable the addition of new ingest data types.

#### Attributes:

**myDBAccess** - The pointer to the DBAccess object class.

Data Type:InDBAccess\*

Privilege:Private

Default Value:

**ourDataTypeTemplateList** - List of all the valid Data Types.

Data Type:RWTPtrOrderedVector<InTDbDataTypeTemplate

Privilege:Private

Default Value:static

**ourInstance** - Single instance of this object class.

Data Type:InDataTypeTemplate\*

Privilege:Private

Default Value:static

#### Operations:

**GetDSArchiveUR** - GetDSArchiveUR for a particular DataType.

Arguments:const RWCString dataType, RWCString& DSArchiveUR

Return Type:EcUtStatus

Privilege:Public

**GetDSResourceUR** - GetDSResourceUR for a particular DataType.

Arguments:const RWCString dataType, RWCString& DSResourceUR

Return Type:EcUtStatus

Privilege:Public

**GetDataTypeInfo** - Get URs for a particular DataType.

Arguments:const RWCString datatype, RWCString& DSArchiveUR, RWCString& reqMgrUR, RWCString& DSResourceUR

Return Type:EcUtStatus

Privilege:Public

**GetDataTypes** - Retrieve all Ingest data types.

Arguments:RWTPtrOrderedVector<RWCString>& dataTypes

Return Type:EcUtStatus

Privilege:Public

**GetReqMgrUR** - Get ReqMgrUR for a particular DataType.

Arguments:const RWCString dataType, RWCString& reqMgrUR

Return Type:EcUtStatus

Privilege:Public

**InDataTypeTemplate** - This is the object class constructor.

Arguments:InDBAccess\* DBAccess

Return Type:Void

Privilege:Protected

**Instance** - Return the single instance of this class.

Arguments:InDBAccess\* DBAccess, EcUtStatus&status

Return Type:static InDataTypeTemplate\*

Privilege:Public

**UpdateUR** - Update a UR for a particular DataType. Phase 3 capability.

Arguments:const RWCString dataType, const RWCStringUR, const InTBdURTypeOfUR

Return Type:EcUtStatus

Privilege:Public

**~InDataTypeTemplate**

Arguments:EcTVoid

Return Type:Void

Privilege:Public

## **Associations:**

The InDataTypeTemplate class has associations with the following classes:

None

### 4.3.18 InExternalDataProviderInfo Class

Parent Class:InDBAccess

Public:No

Distributed Object:No

Persistent Class:True

Purpose and Description:

Persistent thresholds on an External Data Provider basis for limits on Ingest request traffic, data volumes, and data transfer retries.

#### Attributes:

**myExternalDataProvider** - Identifier of the external data provider (e.g., TSDIS) that supplies ingest requests.

Data Type:RWString

Privilege:Private

Default Value:

**myIngestMode** - Indicates if the external data provider uses automatic network ingest, polling files ingest, polling delivery record ingest, interactive ingest, or hard media ingest.

Data Type:RWString

Privilege:Private

Default Value:

**myState** - Current state of External Data Provider's request processing. Options include active or suspended.

Data Type:RWString

Privilege:Private

Default Value:

#### Operations:

**GetState** - Returns the current state of the specified External Data Provider's request processing.

Arguments:RWString DataProviderID, RWString State

Return Type:EcUtStatus

Privilege:Public

**InExternalDataProviderInfo** - Constructor of the object class.

Arguments:EcTvoid

Return Type:Void

Privilege:Public

**UpdateState** - Update state of External Data Provider's request processing. Options include active or suspended.

Arguments:RWString State

Return Type:EcUtStatus

Privilege:Public

**~InExternalDataProviderInfo** - Destructor of the object class.

Arguments:EcTvoid

Return Type:Void

Privilege:Public

#### **Associations:**

The InExternalDataProviderInfo class has associations with the following classes:

None

#### **4.3.19 InFDFData Class**

Parent Class:InScienceData

Public:No

Distributed Object:No

Purpose and Description:

This class provides services to preprocess FDF data into acceptable data server format.

#### **Attributes:**

All Attributes inherited from parent class

#### **Operations:**

**InFDFData** - This is the constructor service.

Arguments:

Return Type:Void

Privilege:Public

**Preprocess** - Performs data preprocessing on the corresponding data.

Arguments:EcTvoid

Return Type:EcTInt

Privilege:Public

**~InFDFData** - This is the destructor.

Arguments:

Return Type:Void

Privilege:Public

#### **Associations:**

The InFDFData class has associations with the following classes:

None

#### **4.3.20 InFile Class**

Parent Class:Not Applicable

Public:No

Distributed Object:No

Persistent Class:

Purpose and Description:

Instantiates an ingested file on available storage space by collaborating with the DsStResource object class services (described in the Data Server Subsystem section of this document). The InFile object class also performs the file size and file existence checks.

#### **Attributes:**

**myErrorStatus** - Contains the status of the ingest file.

Data Type:int

Privilege:Private

Default Value:

**myFileId** - The unique identifier of the ingest file.

Data Type:char \*

Privilege:Private

Default Value:

**myFileLocation** - Identifies the location the file resides.

Data Type:char \*

Privilege:Private

Default Value:

**myFileType** - Identifies the file type(e.g., metadata file, science data, calibration) of the ingest file.

Data Type:char \*

Privilege:Private

Default Value:

**myFileVolume** - The data volume of the ingest file.

Data Type:int

Privilege:Private

Default Value:

**myNodeName** - The host name where the file resides.

Data Type:char \*

Privilege:Private

Default Value:

## **Operations:**

**GetErrorStatus** - Returns the status of the ingest file.

Arguments:

Return Type:Void

Privilege:Public

**GetFileId** - Returns the file name of the ingest file.

Arguments:

Return Type:Void

Privilege:Public

**GetFileLocation** - Returns the file location of the ingest file.

Arguments:

Return Type:Void

Privilege:Public

**GetFileType** - Returns the file type (e.g., metadata file, science data, calibration data) of the ingest file.

Arguments:

Return Type:Void

Privilege:Public

**GetFileVolume** - Returns the size of the ingest file.

Arguments:

Return Type:Void

Privilege:Public

**GetNodeName** - Returns the host name of the ingest file.

Arguments:

Return Type:Void

Privilege:Public

**Transfer** - Initiates transfer of the ingest file.

Arguments:

Return Type:Void

Privilege:Public

#### **Associations:**

The InFile class has associations with the following classes:

Class: InDataPreprocessTask processes

### **4.3.21 InFileTypeTemplate Class**

Parent Class:InDBAccess

Public:No

Distributed Object:Yes

Persistent Class:True

Purpose and Description:

This class is responsible for storing information that categorizes each ingest file type (e.g. metadata vs science data). This information is used by the InDataType Class to create the appropriate specializations of the InMetadata and InScienceData base classes. The class will contain the necessary information on how to process each specific file type.

#### **Attributes:**

**myCmd** - This attribute specifies the Sybase command pointer.

Data Type:CS\_COMMAND

Privilege:Private

Default Value:

**myCtr**

Data Type:EcTInt

Privilege:Private

Default Value:

**myDataType** - This attribute specifies the data type (e.g. CER00, LIS00) for the file type template object.

Data Type:RWCString\*

Privilege:Private

Default Value:

Constraints:

Non Persistent Flag:False

**myFTarray** - This attribute specifies the structure which contains information on the constituent file types of a specific data type.

Data Type:RWTPtrOrderedVector<RWCString>

Privilege:Private

Default Value:

**myFileType** - This attribute specifies the file type (e.g. metadata, science) of the file type template object.

Data Type:RWCString

Privilege:Private

Default Value:

Constraints:

Non Persistent Flag:False

**myFileTypeInfo** - This attribute specifies the structure which contains information pertaining to a specific file type.

Data Type:FileTypeInfo\*

Privilege:Private

Default Value:

**myFileanme\_vector** - This attribute specifies the ordered vector which contains a list of file type structures.

Data Type:RWTPtrOrderedVector<FileTypeInfo>

Privilege:Private

Default Value:

**myNumCols** - This attribute specifies the number of columns in the result set.

Data Type:CS\_INT

Privilege:Private

Default Value:

**myRetcode** - This attribute specifies the sybase return code.

Data Type:CS\_RETCODE

Privilege:Private

Default Value:

**mySQLCmd** - This attribute defines the SQL command pointer.

Data Type:RWCString\*

Privilege:Private

Default Value:

## Operations:

**DeleteSession** - Deletes file type information associated with the given session id.

Arguments:int SessionID

**GetDTInfo** - This service provides a list of the constituent file types for a specific data type.

Arguments:RWTPtrOrderedVector<RWCString>\*\* VecHolder

Return Type:EcUtStatus

Privilege:Public

**GetFTInfo** - This member function provides file type processing information for a specific file type.

Arguments:RWCString FileType, FileTypeInfo\* FTHolder

Return Type:EcUtStatus

Privilege:Public

**InFileTypeTemplate** - This is the constructor service.

Arguments:RWCString DataType

Return Type:void

Privilege:Public

**SearchSession** - Select file type information associated with the given session id.

Arguments:int SessionID

**~InFileTypeTemplate** - This is the destructor service.

Arguments:void

Return Type:void

Privilege:Public

### **Associations:**

The InFileTypeTemplate class has associations with the following classes:

Class: InDataPreprocessTask guides

### **4.3.22 InGRIBData Class**

Parent Class: InScienceData

Public: No

Distributed Object: No

Purpose and Description:

This class will provide services to preprocess science data in GRIB format.

### **Attributes:**

All Attributes inherited from parent class

### **Operations:**

**InGRIBData** - This is the constructor service.

Arguments:

Return Type: Void

Privilege: Public

**Preprocess** - This operation converts data in GRIB format to HDF format.

Arguments:

Return Type: EcUtStatus

Privilege: Public

**~InGRIBdata** - This is the destructor service.

Arguments:

Return Type: Void

Privilege: Public

### **Associations:**

The InGRIBData class has associations with the following classes:

None

### 4.3.23 InGUISession Class

Parent Class:EcPFManagedServer  
Public:No  
Distributed Object:No  
Persistent Class:  
Purpose and Description:  
Main for GUI task.

#### Attributes:

All Attributes inherited from parent class

#### Operations:

**main** - This is the main driver for the Ingest GUI.  
Arguments:  
Return Type:Void  
Privilege:Public

#### Associations:

The InGUISession class has associations with the following classes:  
Class: InIngestMainWindow

### 4.3.24 InGranuleAsync\_CB Class

Parent Class:Not Applicable  
Public:No  
Distributed Object:No  
Purpose and Description:  
Handles all communication messages to and from InGranuleAsync\_S object for a single request granule.

#### Attributes:

None

## Operations:

**CancelB** - This routine overloads EcSrAsyncRequest\_c::Cancel. It will send a message to the InGranuleAysnc\_S object to cancel remote processing.

Arguments:EcTvoid

Return Type:EcUtStatus

Privilege:Public

**CompleteB** - Overloads SRF complete. Deletes granule from RWlist GranuleList.

Arguments:EcTvoid

Return Type:EcUtStatus

Privilege:Public

**InGranuleAsync\_CB** - Constructor. Constructed by InGranuleServer\_S via SRF.

Arguments:InGranuleAsyncRequest\_S &ctorMSG

Return Type:Void

Privilege:Public

**SuspendB** - Overloads SRF suspend. Since at the granule level a suspend and a resume are handled the same, this function calls InGranuleAsync\_C::Cancel.

Arguments:EcTvoid

Return Type:EcUtStatus

Privilege:Public

**~InGranuleAsync\_C**

Arguments:EcTvoid

Return Type:Void

Privilege:Public

Associations:

The InGranuleAsync\_CB class has associations with the following classes:

Class: InGranuleServer\_SB Constructs

Class: InGranuleAsync\_SB ExchangesStateMessages

### 4.3.25 InGranuleAsync\_SB Class

Parent Class:Not Applicable

Public:No

Distributed Object:Yes

Purpose and Description:

This is an object constructed by InGranuleServer\_S to handle the processing and messages for a single granule. Processing includes Granule file transfer, preprocessing and Data Server insertion.

#### Attributes:

**myClientUR** - UR of corresponding client object.

Data Type:EctUR

Privilege:Private

Default Value:

**myDanGranuleFileName** - File name of DAN Granule File. The DAN granule file contains DAN information for a single Granule

Data Type:RWString

Privilege:Private

Default Value:

**myGranuleID** - Granule ID of granule within the request. Used for data base access

Data Type:EctInt

Privilege:Private

Default Value:

**myRequestID** - Request ID used for order tracking and data base access purposes.

Data Type:EctInt

Privilege:Private

Default Value:

**myResumeFlag** - Flag indicating if the granule is being restarted for recovery or resume purposes.

Data Type:EctBoolean

Privilege:Private

Default Value:

## Operations:

**Cancel** - Based on current state of granule, sends cancel request on to Data Server resource mananagement or data insertion software.

Arguments:EcInt RequestID, EcIntGranuleID

Return Type:EcUtStatus

Privilege:Public

**CheckGranuleStateB** - Called to check state attribute of object to determine if granule processing has been suspended or cancelled.

Arguments:EcTInt RequestID, EcTInt GranuleED, EcTInt ResultingAction

Return Type:EcUtStatus

Privilege:Public

**ExecuteB** - Executed from SRF. Controls main granule processing including file transfer, preprocessing and data insertion.

Arguments:EcTVoid

Return Type:EcUtStatus

Privilege:Public

**InGranuleAsync\_SB** - This is the object class constructor.

Arguments:RWString &DANGranuleFileName, EcTBoolean ResumeFlag, EcTInt RequestID, EcTInt GranuleID, InGranuleAsync\_c\* &async

Return Type:Void

Privilege:Public

**SetPriority B** - Sets the priority level for the corresponding granule.

Arguments:EcInt RequestID, EcIntGranuleID

Return Type:EcUtStatus

Privilege:Public

**SuspendB** - Based on current state of granule, sends suspend request onto either Data Server Resource manager or Data Server Insert software.

Arguments:EcInt RequestID, EcIntGranuleID

Return Type:EcUtStatus

Privilege:Public

**~InGranuleAsync\_SB** - This is the object class destructor.

Arguments:EcTVoid

Return Type:Void

Privilege:Public

Associations:

The InGranuleAsync\_SB class has associations with the following classes:

Class: InGranuleServer\_SB Constructs  
Class: InGranuleAsync\_CB ExchangesStateMessages  
Class: InDataPreprocessTask Invokes  
Class: InDataServerInsertionTask Invokes  
Class: InDataTransferTask Invokes

**4.3.26 InGranuleMessageB Class** - This class is the Ingest message class defining the granule Sever\_C to InGranuleServer\_S via SRF.

Parent Class:Not Applicable  
Public:No  
Distributed Object:No  
Purpose and Description:

**Attributes:**

**myMessageStringB** - Message requesting remote granule processing  
Data Type:EcsStreamable  
Privilege:Private  
Default Value:

**Operations:**

**InGranuleMessageB**

Arguments:RWString GranuleFileName, EcRequest OrderID, EcTBoolean ResumeFlag,  
EcTInt RequestID, EcTInt GranuleID

**~InGranuleMessageB**

Arguments:EcTVoid  
Return Type:Void  
Privilege:Public

Associations:

The InGranuleMessageB class has associations with the following classes:  
Class: InGranuleServer\_CB IsInvokedBy

#### 4.3.27 InGranuleServer\_CB Class

Parent Class:Not Applicable

##### Attributes:

None

##### Operations:

###### **InGranuleServer\_CB**

Arguments:EcTvoid

Return Type:Void

Privilege:Public

**ProcessGranuleB** - Starts the ingest processing for the corresponding granule.

Arguments:RWString &DANGranulefileName, EcTBoolean ResumeFlag, EcTint RequestID, EcTint GranuleID

###### **~InGranuleServer\_CB**

Arguments:EcTvoid

Return Type:Void

Privilege:Public

Associations:

The InGranuleServer\_CB class has associations with the following classes:

Class: InRequest

Class: InGranuleServer\_SB Invokes

Class: InGranuleMessageB IsInvokedBy

#### 4.3.28 InGranuleServer\_SB Class

Parent Class:EcPFManagedServer

Public:No

Distributed Object:Yes

Purpose and Description:

Remote granule server. This is a standalone executable that acts as an object factory. It will run on any Data Server instance where granule processing will be performed. This is an SRF Server.

## Attributes:

**myClientURB** - Universal Reference of corresponding granule server client

Data Type:EcTUR

Privilege:Private

Default Value:

**myLocalServerB** - The Universal Reference (UR) of the granule server.

Data Type:EcTUR

Privilege:Private

Default Value:

## Operations:

### **InGranuleServer\_SB**

Arguments:EcURUr myUR, EcSrRequestDispatcher\* localServer

Return Type:EcUtStatus

Privilege:Public

**ProcessRequestB** - Receives granule process requests from InGranuleServer\_C. Constructs local InGranuleAsync\_S and remote InGranuleAsync\_C objects to handle actual granules processing and all subsequent granule traffic.

Arguments:RWString &DANGranuleFileName, EcTBoolean ResumeFlag, EcTInt RequestID, EcTInt GranuleID, InGranuleAsync-c\* &async

Return Type:EcUtStatus

Privilege:Public

**ReceiveMsgRAcceptanceB** - Receives granule processing messages from InGranuleServer\_C. Message unpacking is handled by calling SRF functionality. Checks message classID to verify it is a process granule message then calls ProcessGranule.

Arguments:EcCsMSG\* newMessage, EcURUr\* clientUR

Return Type:EcUtStatus

Privilege:Public

### Associations:

The InGranuleServer\_SB class has associations with the following classes:

Class: InGranuleAsync\_CB Constructs

Class: InGranuleAsync\_SB Constructs

Class: InGranuleServer\_SB

Class: InGranuleServer\_CB Invokes

### 4.3.29 InHDFMetadata Class

Parent Class:InMetadata

Public:No

Distributed Object:No

Purpose and Description:

This class will provide services to preprocess HDF metadata.

#### Attributes:

All Attributes inherited from parent class

#### Operations:

**InHDFMetadata** - This is the constructor service.

Arguments:

Return Type:Void

Privilege:Public

**Preprocess** - This operation will extract values from HDF input files. It will accesss the correct MCFs and interact with the Metadata Toolkit to produce a modified metadata file which is acceptable to the Data Server Subsystem.

Arguments:EcTVoid

Return Type:EcTInt

Privilege:Public

**~InHDFMetadata** - This is the destructor service.

Arguments:

Return Type:Void

Privilege:Public

#### Associations:

The InHDFMetadata class has associations with the following classes:

None

### 4.3.30 InHistoryLog Class

Parent Class: InIngestMainWindow

Public: No

Distributed Object: No

Persistent Class: True

Purpose and Description:

Contains ingest history information. The object class provides services retrieve ingest events from the log.

#### Attributes:

All Attributes inherited from parent class

#### Operations:

**ProcessRequest** - Retrieves ingest events from the history log based on the user specified criteria and displays the events to the user's screen.

Arguments: InTHLSearchCriteria SearchCriteria, EcTInt DetailLevel, EcTInt\* NumOfRows, RWCString HistLogFile

Return Type: EcUtStatus

Privilege: Public

**ProcessRequest** - Retrieves ingest events from the history log based on the user specified Request ID and displays the events to the user's screen.

Arguments: EcTInt ReqID, EcTInt\* NumOfRows, RWCString HistLogFile

Return Type: EcUtStatus

Privilege: Public

#### Associations:

The InHistoryLog class has associations with the following classes:

Class: InRequestSummaryData Accesses

Class: InRequestSummaryHeader Accesses

### 4.3.31 InIngestMainWindow Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

This is the abstract parent object class from which all of the Ingest GUI object classes inherits.

#### Attributes:

None

#### Operations:

**CheckPrivileges** - Verifies that the user has privilege to access the Ingest history log.

Arguments:RWString UserID

Return Type:EcTInt

Privilege:Public

**ReceiveMsg** - Reads the user spcified inputs from the screen.

Arguments:EcTVoid

Return Type:EcTInt

Privilege:Public

**SendMsg** - Displays message to the user's screen.

Arguments:EcTVoid

Return Type:EcTInt

Privilege:Public

#### Associations:

The InIngestMainWindow class has associations with the following classes:

Class: InGUISession

### 4.3.32 InInteractiveIngestB Class

Parent Class:EcPFManagedServer

Public:No

Distributed Object:No

Purpose and Description:

#### Attributes:

**myDANIDB** - ID of Dan file created or supplied to Interactive Ingest.

Data Type:RWString

Privilege:Private

Default Value:

**myUserNameB** - Interactive user name. Used to access user's profile to locate local directories and retrieve Email address.

Data Type:RWString

Privilege:Private

Default Value:

#### Operations:

**BuildDANB** - Builds DAN file from arguments, stores DAN in ECS controlled User area.

Returns status.

Arguments:RWList FileList, RWList DataType, RWList DANHeader

Return Type:EcUtStatus

Privilege:Public

**DetermineFileSizeB** - Uses UNIX system services to determine file size of local to ECS files.

Arguments:RWString FileName, RWString FileLocation

Return Type:EcUtStatus

Privilege:Public

**ExecuteScriptB** - Script main driver.

Arguments:EcTVoid

Return Type:EctVoid

Privilege:Public

**GetDANHeaderB** - Process DAN header information supplied via HTML on the DANHeaderForm

Arguments:RWList HeaderInformation

Return Type:EcUtStatus

Privilege:Public

**GetDataTypeInfoB** - Processes data type information supplied via HTML from the DataTypeInformationForm.

Arguments:RWList DataTypeInformation

Return Type:EcUtStatus

Privilege:Public

**ProcessDANIngestB** - Process DAN file name submitted by user via HTML DANIngestForm. Calls InDAN to construct DAN object. Call InDAN::Checkpoint to checkpoint DAN information in Ingest data base. Sends request to InRequestManage for processing.

Arguments:EcTVoid

Return Type:EcUtStatus

Privilege:Public

**ProcessDirSpecMultiDANB** - Drives the processing of a single directory containing multiple granules where each granule is distinguished by having the same base file name. Creates a list of files, partitions into granules, gets data type and file type, builds DAN and sends DAN to InRequestManager for processing.

Arguments:RWString DirectoryName, RWString DataType, RWList DANHeader

Return Type:EcUtStatus

Privilege:Public

**ProcessDirSpecSingleDANB** - Using UNIX system calls, determines the file content of a single directory. Displays, via HTML RetrieveFileTypes Form, the name of each file in the directory. The form indicates that the user should fill in File Type for each File displayed. Calls BuildDan operation and sends request to InRequestManage.

Arguments:RWString DirectoryName, RWString DataType, RWList DANHeader

Return Type:EcUtStatus

Privilege:Public

**ProcessInteractiveDANSpecB** - Controls the forms session with the user to interactively build a DAN request.

Arguments:EcTVoid

Return Type:EcUtStatus

Privilege:Public

**ProcessUserNameB** - Creates User Profile object. Returns error status if user can not be located in User Profile.

Arguments:RWString UserName, EcUserProfile& theUser

Return Type:EcUtStatus

Privilege:Public

**WriteDANGroupB** - Called by InBuildDAN. Writes DAN file group information in PVL format.

Arguments:RWList FileList, RWList DataType

Return Type:EcUtStatus

Privilege:Public

**WriteDANHeaderB** - Write DAN header information to DAN file in PVL format.

Arguments:RWList HeaderList

Return Type:EcUtStatus

Privilege:Public

Associations:

The InInteractiveIngestB class has associations with the following classes:

Class: InRequestManager\_C SendsRequestTo

Class: InRequestManager\_S SendsRequestTo

### 4.3.33 InLongDAA Class

Parent Class:InMessage

Public:No

Distributed Object:No

Purpose and Description:

This object class populates the long DAA (DAN Acknowledgement) data message to be sent to the external Client after the receipt of the DAN.

#### Attributes:

**myLongDAA** - This is the DAN Acknowledgement data message in detailed format.

Data Type:LongDAAMsg

Privilege:Private

Default Value:

### **Operations:**

**FillDAA** - This function will package a long DAA message for the DAN acknowledgement.

Arguments: int Status[], char \*DataType[], char \*Descriptor, int FileGroupCount, int DANSeqNo

Return Type: Void

Privilege: Public

### **Associations:**

The InLongDAA class has associations with the following classes:

None

## **4.3.34 InLongDDN Class**

Parent Class: InMessage

Public: No

Distributed Object: No

Purpose and Description:

This object class populates the long DDN (Data Delivery Notice) data message to be sent to the external Client after the data is archived.

### **Attributes:**

**myLongDDN** - This is the long DDN (Data Delivery Notice) data message.

Data Type: LongDDNmsg

Privilege: Private

Default Value:

### **Operations:**

**FillDDN** - Packages the DDN data message with given inputs.

Arguments: int Status[], char \*Direcotry, char \*FileId, int FileCount, int DANSeqNo

Return Type: Void

Privilege: Public

## **Associations:**

The InLongDDN class has associations with the following classes:

Class: InRequest builds

### **4.3.35 InMediaIngest Class**

Parent Class: InIngestMainWindow

Public: No

Distributed Object: No

Persistent Class:

Purpose and Description:

Provides operations personnel the capability to perform physical media ingest via the GUI interface. This is a derived object class from the InGUISession object class. It inherits all data and service members provided by the InGUISession.

## **Attributes:**

**myUserName** - The user name of the operator requesting the Media Ingest service.

Data Type: RWCString

Privilege: Private

Default Value:

## **Operations:**

**AllocDevice** - Allocates for an available specified peripheral device.

Arguments: EcTInt MediaType

Return Type: EcUtStatus

Privilege: Public

**AllocStDisk** - Allocates for an available staging disk.

Arguments: void

Return Type: EcUtStatus

Privilege: Public

**CheckPrivilege** - Verifies that the operator and the media provider has ingest privilege.

Arguments: char \*UserName

Return Type: int

Privilege: Public

**Copy** - Down-loads data from tape to staging.

Arguments:void

Return Type:EcUtStatus

Privilege:Public

**DeallocDevice** - Deallocates the peripheral device.

Arguments:void

Return Type:EcUtStatus

Privilege:Public

**DeallocStDisk** - Deallocates the staging disk.

Arguments:void

Return Type:EcUtStatus

Privilege:Public

**DismountMedia** - Dismount the tape.

Arguments:void

Return Type:EcUtStatus

Privilege:Public

**GetBarCodeB** - Retrieves the bar code information from the media tape.

Arguments:RWString BarCode

Return Type:Void

Privilege:Public

**GetDDRFile** - Retrieves the Data Delivery Record (DDR) file from the tape.

Arguments:EcTInt DDRLocation, EcTInt MediaType, RWString DDRFileName

Return Type:EcUtStatus

Privilege:Public

**InsertRequestB** - Performs the Data Delivery Record file validation and inserts the information into the Data Base.

Arguments:EcTInt\* ReqID

Return Type:EcUtStatus

Privilege:Public

**MountMedia** - Mounts the tape.

Arguments:RWString MediaVolID

Return Type:EcUtStatus

Privilege:Public

**ProcessRequest** - Submits the request to the Ingest Request Manager for ingest processing.

Arguments:void

Return Type:Void

Privilege:Public

**ProcessRequestB** - Submits the request to the Ingest Request Manager for ingest processing.

Arguments:EcTInt ReqID

Return Type:EcUtStatus

Privilege:Public

#### **Associations:**

The InMediaIngest class has associations with the following classes:

Class: InRequestManager\_S SendsReqeustTo

Class: DsStResourceProvider uses

### **4.3.36 InMessage Class**

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

Contains data messages that interchanges between the external Client and ECS/Ingest.

#### **Attributes:**

None

#### **Operations:**

**GetMsgLength** - Extracts and returns the message type from the data message.

Arguments:char \*MsgPtr

Return Type:Void

Privilege:Public

**InMessage** - This is the constructor for the InMessage object class.

Arguments:

Return Type:Void

Privilege:Public

## Associations:

The InMessage class has associations with the following classes:

Class: InSession Receives/Sends - The InSession object interfaces with the InMessage object to access data messages that are interchanged between Ingest and the external Client.

### 4.3.37 InMetadata Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

This is an abstract class.

## Attributes:

**myFileTypeRow** - This attribute a structure which contains preprocessing profile information for a specific data type/file type.

Data Type:FileTypeInfo\*

Privilege:Private

Default Value:

**myInputFile** - This attribute references the raw metadata input file.

Data Type:RWCString

Privilege:Private

Default Value:

**myTargetMCF** - This attribute references the target MCF obtain from the Data Server Subsystem.

Data Type:RWCString

Privilege:Private

Default Value:

## Operations:

**Preprocess** - This is a pure virtual member function.

Arguments:Abstract

Return Type:Void

Privilege:Public

## Associations:

The InMetadata class has associations with the following classes:

Class: InSourceMCF defines - The InSourceMCF class defines the InMetadata class by providing format information on input metadata files.

Class: DsCIDescriptor guides - The DsCIDescriptor class guides the InMetadata class by providing services to access target MCFs and validate metadata files.

InDataPreprocessTask (Aggregation)

### 4.3.38 InNextAvailableID Class

Parent Class: InDBAccess

## Attributes:

**myDBAccess** - The pointer to the instance of the InDBAccess class used to connect to the database.

Data Type: InDBAccess\*

Privilege: Private

Default Value:

**myRequestID** - The next available RequestID.

Data Type: int

Privilege: Private

Default Value:

## Operations:

**GetNewReqID** - Get a new RequestID.

Arguments: int&

Return Type: EcUtStatus

Privilege: Public

**GetReqID** - Return the RequestID.

Arguments: void

Return Type: int

Privilege: Public

**InNextAvailableID** - Alternate constructor.

Arguments:InDBAccess\*

Return Type:Void

Privilege:Public

**~InNextAvailableID** - Destructor.

Arguments:void

Return Type:Void

Privilege:Public

#### **Associations:**

The InNextAvailableID class has associations with the following classes:

None

#### **4.3.39 InPVMetadata Class**

Parent Class:InMetadata

Public:No

Distributed Object:No

Purpose and Description:

This class will provides services to preprocess Parameter-Value metadata.

#### **Attributes:**

**myLineDelimiter** - This attribute will define the symbol used to indicate the end of a parameter-value metadata statement.

Data Type:RWCString

Privilege:Private

Default Value:

**myParameterDelimiter** - This attribute will define the symbol used to delimit the parameter portion of a parameter-value metadata statement.

Data Type:RWCString

Privilege:Private

Default Value:

**mySeparator** - This attribute will define the symbol used to separate the parameter from the value in the parameter-value metadata statement

Data Type:RWCString

Privilege:Private

Default Value:

**myValueDelimiter** - This attribute will define the symbol used to delimit the value part of a parameter-value metadata statement.

Data Type:RWCString

Privilege:Private

Default Value:

## Operations:

**InPVMetadata** - This is the constructor service.

Arguments:

**Preprocess** - This operation will extract values from input parameter-value files based on delimiters and parameter names. It will access the correct MCFs and interact with the InMetadata\_Tool Class to produce a metadata file which is acceptable to the data server subsystem.

Arguments:

Return Type:EcTInt

Privilege:Public

**~InPVMetadata**

Arguments:

Return Type:Void

Privilege:Public

## Associations:

The InPVMetadata class has associations with the following classes:

None

#### 4.3.40 InPollingIngestSession Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Persistent Class:

Purpose and Description:

This object class does not have any control link with the external interface (i.e, no physical stimulus provided from external source). It is a persistent object class which is configured to wake up at a tunable period of time to detect existence of ingest files at a designated location; the location could either be external or local in ECS. If files are detected, the object class will instantiate the InPollingIngestRequest object class and add it to the InRequestList to be processed. The InPollingIngestSession object class is derived from the InSession. It inherits all the data and service members provided by the InSession object class.

##### Attributes:

**mtDataTypeList** - List of Data Types associated with files to be ingested.

Data Type:char\*\*

Privilege:Private

Default Value:

**myDataProvider** - Indicates the data provider of the files to be polled.

Data Type:char\*

Privilege:Private

Default Value:

**myDataTypeCount** - Specifies the number of data types associated with data files.

Data Type:int

Privilege:Private

Default Value:

**myExprDate** - Indicates how long the data will remain available.

Data Type:char\*

Privilege:Private

Default Value:

**myFtpFileList** - List of files (old and new) in the currently polling directory.

Data Type:InTPoList\*

Privilege:Private

Default Value:

**myHostNameList** - List of host names corresponding to the data types to be polled.

Data Type:char\*\*

Privilege:Private

Default Value:

**myOldFiles** - List of file names that contains the names of the previously ingested files.

Data Type:char\*

Privilege:Private

Default Value:

**myPollingTimer** - The time period which indicates how often the Polling Ingest Session should check for the existence of ingest files for ingest processing.

Data Type:int

Privilege:Private

Default Value:

**mySourceDirectoryList** - List of source directories where the files are to be ingested.

Data Type:char\*

Privilege:Private

Default Value:

## Operations:

**CheckFiles** - Detects for new files from the specified directory

Arguments:char \*HostName, char \*Dir, InTPoList \*\*NewFileList, int \*NewFileCount

Return Type:int

Privilege:Public

**CleanupDirectory** - Performs the cleanup processing after the ingest is complete

Arguments:InTDataTypeList \*DTList, int DTCount

Return Type:int

Privilege:Public

**DeleteItem** - Deletes an item from the linked list.

Arguments:InTPoList\*\* List, char \*Data

Return Type:void

Privilege:Private

**DeleteList** - Destroys the linked list.

Arguments:InTPoList\*\* List

Return Type:void

Privilege:Private

**DeliverResponse** - Generates the ingest response pertaining to the ingest polling process in a file.

Arguments:char \*ResponseFile

Return Type:int

Privilege:Public

**InPollingIngestSession**

Arguments:int argc, char \*\*argv, int status

Return Type:Void

Privilege:Public

**InitList** - Initializes the linked list.

Arguments:InTPoList\*\*

Return Type:void

Privilege:Private

**InsertList** - Inserts an item into the linked list.

Arguments:InTPoList\*\* List, char \*Data

Return Type:void

Privilege:Private

**IsFileNew** - This function will determine if the specified file is a new file by checking the file against the names of previously ingested files.

Arguments:char \*FileName

Return Type:int

Privilege:Public

**IsItemInList** - Searches for an item in the linked list.

Arguments:InTPoList\*\* List, char \*Data

Return Type:int

Privilege:Private

**ProcessRequest** - Creates a new Ingest request using the Delivery Record file as input and adds the request to the In Request List for processing.

Arguments:char \* DANFile

Return Type:int

Privilege:Public

**ProcessRequest** - Begins the ingest processing. The processing includes data transfer, data preprocessing, and Data Server insertion.

Arguments:InTDataTypeList \*DTList, int DTCount

Return Type:int

Privilege:Public

**Associations:**

The InPollingIngestSession class has associations with the following classes:

Class: EcPFManagedServer

Class: InRequestManager\_C SendsRequestTo

**4.3.41 InPollingThreshold Class**

Parent Class:Not Applicable

Public:No

Distributed Object:No

Persistent Class:True

Purpose and Description:

This is a persistent object class that defines thresholds for the Ingest Polling Interface.

**Attributes:**

None

**Operations:****GetPollingThresholds**

Arguments:char\* ExternalDataProviders, int PollingTimers, int rowcount

**Associations:**

The InPollingThreshold class has associations with the following classes:

Class: InRequestManager accesses

**4.3.42 InReformatData Class**

Parent Class:InScienceData

Public:No

Distributed Object:No

Purpose and Description:

This class provides services to preprocess data which is not in an ECS compatible format.

**Attributes:**

All Attributes inherited from parent class

**Operations:**

**InReformatData** - This is the constructor service.

Arguments:

Return Type:Void

Privilege:Public

**Preprocess** - This operation will reformat input science data files which are not in an acceptable ECS data format. The reformatting includes byte swapping and other functions to resolve platform incompatibilities.

Arguments:

Return Type:EcTInt

Privilege:Public

**~InReformatData**

Arguments:

Return Type:Void

Privilege:Public

**Associations:**

The InReformatData class has associations with the following classes:

None

**4.3.43 InRequest Class**

Parent Class:Not Applicable

Public:No

Distributed Object:No

Persistent Class:

Purpose and Description:

Contains information provided by a requestor/external interface requesting ingest of data. The object class has the responsibility to perform basic request component checking and to assign a unique identifier for the new ingest request.

## Attributes:

**myAggregateLength** - Total volume of data (in bytes) to be ingested based on the given request.

Data Type:int

Privilege:Private

Default Value:

**myDataTypeIdList** - The set of data types associate with the ingest granules.

Data Type:struct \*\*

Privilege:Private

Default Value:

**myExpirationDateTime** - Date/time by which the corresponding ingest request must be completed (i.e., archive insertion complete and response returned to the external data provider).

Data Type:char\*

Privilege:Private

Default Value:

**myExternalDataProvider** - Identifier of the external data source providing data to be ingested into ECS.

Data Type:char\*

Privilege:Private

Default Value:

**myIngestType** - The type of data ingest to be performed (e.g., media ingest, network ingest).

Data Type:char\*

Privilege:Private

Default Value:

**myProcessingEndDateTime** - Ending date/time (in standard ECS date/time format) at which the ingest request processing completed (the time immediately prior to deleting the object in the destructor service).

Data Type:DateTime

Privilege:Private

Default Value:

**myProcessingStartDateTime** - Starting date/time (in standard ECS time format) at which ingest processing began (time of creation of the InRequest object).

Data Type:DateTime

Privilege:Private

Default Value:

**myRequestId** - The information that uniquely identifies an Ingest Request. Request Identifiers are referenced by Status Requests and other Service Requests that are used to monitor or control the execution of Ingest Requests.

Data Type:int

Privilege:Private

Default Value:

**myRequestPriority** - The information that determines the order in which an ingest request will be processed relative to other ingest requests waiting to be processed. The priority is provided by the InExternalDataProvider object class for each external data provider.

Data Type:int

Privilege:Private

Default Value:

**myRequestState** - State of the corresponding ingest request. Values are "Active" and "Complete".

Data Type:char\*

Privilege:Private

Default Value:

**mySequenceId** - The SequenceId identifies each of the control messages for a given request. The sequence number is first extracted from the DAN, then all the control messages (e.g, DAA,DRA,DRR...) need to contain the same sequence number for a given request.

Data Type:int

Privilege:Private

Default Value:

**mySessionId** - The identifier of the session associated with the ingest request.

Data Type:int

Privilege:Private

Default Value:

**myStateChange** - Defines the control state to which the requirement is to be updated

Data Type:RWString

Privilege:Private

Default Value:

**myTotalFileCount** - Total number of files identified for ingest in the request.

Data Type:int

Privilege:Private

Default Value:

## Operations:

**Cancel** - Cancels an ingest request.

Arguments:void

Return Type:int

Privilege:Public

**CancelB** - Calls NotifyRemoteGranules to notify all remotely processing granules that the request is being cancelled.

Arguments:RWString ResultingAction

Return Type:EcUtStatus

Privilege:Public

**ChangeState** - Updates the state of an ingest request.

Arguments:String \*NewState

Return Type:int

Privilege:Private

**CheckpointRequest** - Check points the request to the data base.

Arguments:void

Return Type:int

Privilege:Public

**CheckRequestStateB** - Checks request object attribute myRequestState to determine if request should be cancelled, suspended, resumed, or recovered.

Arguments:EcTInt \*GranuleID, EcTInt ResultingAction

Return Type:EcUtStatus

Privilege:Public

**CheckVolumeThresholdB** - Checks current volume limits for the request to determine if the next granule can be sent to the granule server for processing

Arguments:EcTInt GranuleID, RWString ResultingAction

Return Type:EcUtStatus

Privilege:Public

**DeleteInteractiveIngest** - Handles all clean up necessary for Interactive Ingest Requests. Includes the sending of EMail to the interactive user and the creation of a HTML formatted DDN message.

Arguments:EcTVoid

Return Type:EcUtStatus

Privilege:Public

**DeletePollingSession** - Handles all clean up necessary for Polling Ingest requests.

Arguments:EcTVoid

Return Type:EcUtStatus  
Privilege:Public

**DeleteSession** - Handles all delete functionality necessary for automated network Ingest requests.

Arguments:EcTVoid  
Return Type:EcUtStatus  
Privilege:Public

**FillData** - Puts request related information into the DAN object class.

Arguments:InDAN \*DAN, int \*SessionID  
Return Type:EcUtStatus  
Privilege:Public

**GetGranuleServerURB** - Accesses InDataType Template to get UR for the Ingest Granule Server which will handle the request granule.

Arguments:EcUR &InGranuleServer, RWString& GranuleDataType  
Return Type:EcUtStatus  
Privilege:Public

**GetRequestId** - Assigns an unique identifier for the ingest request.

Arguments:void  
Return Type:int  
Privilege:Private

**GetSessionId** - Returns the identifier of the Session that the ingest request is running.

Arguments:void  
Return Type:int  
Privilege:Public

**GetState** - Returns the state of the corresponding request is in.

Arguments:void  
Return Type:char\*  
Privilege:Private

**InRequest** - Constructor, when a DAN file is supplied.

Arguments:char\* DANfile  
Return Type:int  
Privilege:Public

**InRequest** - Constructor, when a DAN message is supplied.

Arguments:DANmsg \*DANmsgPtr  
Return Type:int  
Privilege:Public

**NotifyRemoteGranuleB** - Loops over all granules in the RW granule list. Calls InGranuleAsync\_C to pass on request state change to remotely processing granules.

Arguments:EctInt MsgType

Return Type:EcUtStatus

Privilege:Public

**ProcessRequestB** - Main driver for request processing. Processes DAN to get list of Granules. Then for every granule in DAN, it checks current Volume thresholds to determine if the granule can be processed or held. Once processing is allowed, the granule request is sent on to the appropriate InGranuleServer\_C. Once all granules have completed processing, the request delete function is called.

Arguments:EctVoid

Return Type:EcUtStatus

Privilege:Public

**RecoverB** - Checks InRequestProcessData data base table to determine if the request granule was completed prior to warm restart.

Arguments:EcTInt GranuleID, RWString ResultingAction

Return Type:EcUtStatus

Privilege:Public

**ResumeB** - Checks InRequestProcessData data base table to determines if the granule was completed prior to the suspend request.

Arguments:EcTInt GranuleID, EcTInt ResultingAction

Return Type:EcUtStatus

Privilege:Public

**SetPriorityB** - Call NotifyRemoteGranules to let all remotely processing granules know that the request priority should be modified.

Arguments:RWString ResultingAction

Return Type:EcUtStatus

Privilege:Public

**SuspendB** - Calls NotifyRemoteGranule operation to notify all remotely processing granules that the request is suspending

Arguments:RWString ResultingAction

Return Type:EcUtStatus

Privilege:Public

## Associations:

The InRequest class has associations with the following classes:

Class: InRequestManager\_S Creates

Class: InDAN Defines

Class: InGranuleServer\_CB

Class: InRequestProcessData IsStoredIn

Class: InRequestManager Manages

Class: InDataServerInsertionTask SendsInsertRequestVia - The InRequest class SendsInsertRequestVia the InDataServerInsertionTask class to initiate the process necessary to insert data into the Data Server Subsystem.

Class: InLongDDN builds

Class: InShortDDN builds

Class: InRequestProcessHeader isStoredIn

Class: InDataPreprocessTask preprocesses

Class: InDataPreprocessList submits - The InRequest class submits InDataPreprocessList in order to identify the files which need to be preprocessed.

### 4.3.44 InRequestController Class

Parent Class: InIngestMainWindow

Public: No

Distributed Object: No

Persistent Class:

Purpose and Description:

Provides authorized operations personnel the capability to update an ongoing ingest request via the GUI interface. The operations personnel could 1) cancel an ingest request, 2) suspend an ingest request, 3) resume an ingest request, or 4) change priority of an ingest request. This is a derived object class from the InGUISession object class. It inherits all the data and services provided by the InGUISession object class.

## Attributes:

**myRequestId** - The identifier of the ingest request that is to be updated (e.g., cancel, suspend, resume, change priority).

Data Type: int

Privilege: Private

Default Value:

**myUpdateType** - Identifies the type of update to be performed on an ingest request. The types of ingest request updates consist of: cancel, suspend, resume, and change priority.

Data Type:int

Privilege:Private

Default Value:

## **Operations:**

**CreateRequestListB** - Searches and returns requests to satisfy the user specified criteria.

Arguments:RWString ControlCommand, RWList ReqeustList

Return Type:Void

Privilege:Public

**ProcessRequest** - Performs the appropriate update service (e.g., cancel, suspend, resume) on the user specified Request ID.

Arguments:EcTInt RequestID, EcTInt ControlType, EcTInt ControlData

Return Type:EcUtStatus

Privilege:Public

**ProcessRequestStateChangeB** - Performs the appropriate control service (e.g., cancel, suspend, resume) based on the user specified information.

Arguments:EcTIntRequestID, RWString StateControlCommand

Return Type:Void

Privilege:Public

**ProcessSearch** - Searches for the request(s) from the data base based on the user specified criteria and displays the retrieved information to the screen.

Arguments:EcTInt SearchType, RWCString SearchValue, EcTInt DetailLevel, EcTInt\* NumOfRows, RWCString ReqInfoFile

Return Type:EcUtStatus

Privilege:Public

**ProcessSearch** - Searches for the user specified request from the data base and displays the request information to the screen.

Arguments:EcTInt RequestID, EcTInt\* NumOfRows, RWCString ReqInfoFile

Return Type:EcUtStatus

Privilege:Public

**ProcessStatusMonitorRequest** - Invokes appropriate services to perform the update service on the specified ingest request. This service overloads the ProcessRequest() service defined the InGUISession object class.

Arguments:int RequestId, int UpdateType

Return Type:Void  
Privilege:Public

#### **Associations:**

The InRequestController class has associations with the following classes:

Class: InRequestProcessHeader Access  
Class: InRequestProcessData Accesses  
Class: InRequestManager\_C SendsControlRequestsTo

### **4.3.45 InRequestFileInfo Class**

Parent Class:InDBAccess

Public:No

Distributed Object:No

Persistent Class:True

Purpose and Description:

Provides checkpoint storage of file information associated with a given data granule in a given ingest request. In the event of request completion, this item is deleted.

#### **Attributes:**

**myDBAccess** - The pointer to the instance of the InDBAccess class used to connect to the database.

Data Type:InDBAccess\*

Privilege:Private

Default Value:

#### **Operations:**

**GetRequestData** - Select the RequestFileInfo information for a given RequestID.

Arguments:const EcTInt requestID,

RWTPtrOrderedVector<InTDReqFileInfo>&fileInfo

Return Type:EcUtStatus

Privilege:Public

#### **InRequestFileInfo**

Arguments:InDBAccess\* DBAccess

Return Type:Void

Privilege:Public

**Insert** - Insert a new record into the RequestFileInfo table.

Arguments:const EcTInt requestID, EcTInt dataGranID, RWCString fileID, RWCString sourceDirID, RWCString targetDirID, EcTInt fileSize, EcTInt recordSize, RWTime beginDateTime

Return Type:EcUtStatus

Privilege:Public

**UpdateFileState** - Update the FileState field for a given file. Phase 3 capability.

Arguments:const EcTInt requestID, const EcTInt dataGranuleID, const RWCString fileID, const RWCString fileState

Return Type:EcUtStatus

Privilege:Public

**UpdateFileStatus** - Update the file status field for a given file.

Arguments:const EcTInt RequestID, const EcTInt DataGranuleID, const RWCString fileID, const EcTInt fileStatus

Return Type: EcUtStatus

Privilege:Public

**UpdateFileStatus** - Update the FileStatus field for a given RequestID and a GranuleID.

Arguments:const EcTInt requestID, const EcTInt dataGranuleID, const EcTInt fileStatus

Return Type:EcUtStatus

Privilege:Public

**~InRequestFileInfo**

Arguments:EcTVoid

Return Type:Void

Privilege:Public

## Associations:

The InRequestFileInfo class has associations with the following classes:

Class: InRequestProcessData

Class: InDataTransferTask ProvidesFileState

### 4.3.46 InRequestManager Class

Parent Class:EcPFManagedServer

Public:No

Distributed Object:No

Persistent Class:

Purpose and Description:

This is a focal object class of the Ingest CI. It coordinates the ingest processing which includes the initiating of the data transfer and the sending of data insertion request to the appropriate Data Server. The object class also tracks and allows updates the ingest thresholds.

#### **Attributes:**

**myCurrentDataVolumeKeep** - Current running total of data volume in active ingest request. Incremented as new InRequest objects are created; decremented when InRequest objects are deleted.

Data Type:int

Privilege:Private

Default Value:

**myCurrentRequestMutex** - Mutex to control the Request access.

Data Type:DCEPthreadMutex

Privilege:Private

Default Value:

**myCurrentRequests** - Keeps a running total of the number of requests currently in the system. Incremented as new InRequest objects are created; decremented when InRequest objects are deleted.

Data Type:int

Privilege:Private

Default Value:

**myDataVolumeThreshold** - This is the system data volume allowed to be processed concurrently.

Data Type:int

Privilege:Private

Default Value:

**myRequestThreshold** - The maximum number of requests allowed to be processed concurrently.

Data Type:int

Privilege:Private

Default Value:

## Operations:

**CreateRequest** - Creates a new InRequest object in a pthread when a DAN file pointer is provided.

Arguments:char\* DANfile

Return Type:DCEObjRefT\*

Privilege:Public

**CreateRequest** - Creates a new InRequest object in a pthread when a DAN message pointer is provided.

Arguments:DANmsg\* DANmsgPtr

Return Type:DCEObjRefT\*

Privilege:Public

**CreateRequest** - Creates a new InRequest object in a pthread when a DAN file pointer is provided.

Arguments:char\* DANfile

Return Type:DCEObjRefT\*

Privilege:Public

**CreateRequest** - Creates a new InRequest object in a pthread when a DAN message pointer is provided.

Arguments:DANmsg\* DANmsgPtr

Return Type:DCEObjRefT\*

Privilege:Public

**DeleteRequest** - Deletes the request from InRequestProcess\* tables.

Arguments:int RequestId

Return Type:int

Privilege:Public

**DeleteRequestRPC** - This RPC invokes the DeleteRequest() service to perform request deletion.

Arguments:int RequestId

Return Type:int

Privilege:Public

**GetCurrentDataVolume** - Gets the value of the current data volume.

Arguments:void

Return Type:int

Privilege:Public

**GetCurrentRequests** - Gets the value of the current total number of ingest requests in the system.

Arguments:void  
Return Type:int  
Privilege:Public

**ProcRequest** - This service is invoked as a thread spawned by CreateRequestRPC().

Arguments:pthread\_addr\_t arg  
Return Type:pthread\_addr\_t  
Privilege:Private

**ProcessStateChangeB** - Requests cancellation, suspension, resumption, and set priority of an existing InRequest object. The InRequest CancelRequest service is invoked. The success of the cancellation request depends on the state of the ongoing request.

Arguments:DCEObjRefT\* ObjReference  
Return Type:int  
Privilege:Public

**RestoreRequestList** - Recovers the InRequestList object after a process or system failure.

Arguments:void  
Return Type:int  
Privilege:Private

**UpdateCurrentDataVolume** - Updates the value of the current running total of data volume requested to be ingested.

Arguments:int UpdateValue  
Return Type:int  
Privilege:Private

**UpdateCurrentRequests** - Updates the current running total for number of ingest requests in the system.

Arguments:int UpdateValue  
Return Type:int  
Privilege:Private

## **Associations:**

The InRequestManager class has associations with the following classes:

Class: InRequest Manages  
Class: InExternalDataProviderThreshold accesses  
Class: InPollingThreshold accesses

#### 4.3.47 InRequestManager\_C Class

Parent Class:Not Applicable

Public:No

Distributed Object:Yes

Purpose and Description:

The client implementation of the distributed InRequestManager object. This client acts as the intermediary to the object factory (InRequestManager\_S).

##### Attributes:

None

##### Operations:

**CreateRequest** - Service to create an InRequest object via a distributed object factory (InRequestManager\_S) when a DAN message is supplied.

Arguments:DANmsg\* DANmsgPtr

Return Type:int

Privilege:Public

**CreateRequest** - Service to create an InRequest object via a distributed object factory (InRequestManager\_S) when a DAN file is supplied.

Arguments:char\* DANfile

Return Type:int

Privilege:Public

**CreateRequest** - Service to create an InRequest object via a distributed object factory (InRequestManager\_S) when a DAN message is supplied.

Arguments:DANmsg\* DANmsgPtr

Return Type:int

Privilege:Public

**CreateRequest** - Service to create an InRequest object via a distributed object factory (InRequestManager\_S) when a DAN file is supplied.

Arguments:char\* DANfile

Return Type:int

Privilege:Public

## **Associations:**

The InRequestManager\_C class has associations with the following classes:

- Class: InSession Invokes
- Class: InSession ReceivesRequestFrom
- Class: InRequestController SendsControlRequestsTo
- Class: InInteractiveIngestB SendsRequestTo
- Class: InPollingIngestSession SendsRequestTo
- Class: InSession SendsRequestTo
- InRequestManager (Aggregation)

### **4.3.48 InRequestManager\_S Class**

Parent Class:Not Applicable

Public:No

Distributed Object:Yes

Purpose and Description:

Implementation of the server (object factory) for InRequestManager.

## **Attributes:**

None

## **Operations:**

**CreateRequest** - Service to create an InRequest\_S object when a DAN message is supplied.

Arguments:DANmsg\* DANmsgPtr

Return Type:int

Privilege:Public

**CreateRequest** - Service to create an InRequest\_S object when a DAN file is supplied.

Arguments:char\* DANfile

Return Type:int

Privilege:Public

## Associations:

The InRequestManager\_S class has associations with the following classes:

Class: InRequest Creates

Class: InMediaIngest SendsRequestTo

Class: InInteractiveIngestB SendsRequestTo

InRequestManager (Aggregation)

### 4.3.49 InRequestProcessData Class

Parent Class: InDBAccess

Public: No

Distributed Object: No

Persistent Class: True

Purpose and Description:

Provides checkpoint storage of data granule processing information associated with a given ingest request. In the event of request completion, this item is deleted.

## Attributes:

**myDBAccess** - The pointer to the instance of the InDBAccess class used to connect to the database.

Data Type: InDBAccess\*

Privilege: Private

Default Value:

## Operations:

**GetGranuleStateForRecoveryB** - This function will call a Sybase stored procedure to determine what the granule state was prior to abort of request processing. The stored procedure will check to see if the granule had been previously restarted. If it had, the stored procedure will return failure status. The stored procedure sets retry flag to 1 to indicate a retry has occurred. The C routine will return ResultingAction based on status of stored procedure.

Arguments: EcTInt ReqeustID, EcTInt GranuleID, RWString ResultingAction

Return Type: EcUtStatus

Privilege: Public

**GetRequestData** - Select RequestProcessData information for the RequestID.

Arguments: const EcTInt requestID, const RWCString filename, EcTInt& rowcount

Return Type:EcUtStatus  
Privilege:Public

**InRequestProcessData**

Arguments:InDBAccess\* DBAccess  
Return Type:Void  
Privilege:Public

**Insert** - Insert a new record into the RequestProccessData table and insert file info into the RequestFileInfo table.

Arguments:const EcTInt requestID, const InTDbReqProcessData\* reqProcessData,const RWTPtrOrderedVector<InTDbReqFileInfo>\* reqFileInfoList  
Return Type:EcUtStatus  
Privilege:Public

**UpdateState** - Update the state for a given DataType.

Arguments:const RWCString dataTypeState, const EcTInt requestID, const EcTInt granuleID, const RWCString dataType const EcTInt timeToCompleteState  
Return Type:EcUtStatus  
Privilege:Public

**~InRequestProcessData**

Arguments:EcTVoid  
Return Type:Void  
Privilege:Public

**Associations:**

The InRequestProcessData class has associations with the following classes:

Class: InRequestController Accesses  
Class: InRequestFileInfo  
Class: InRequestSummaryData IsMovedTo  
Class: InRequestSummaryData IsMovedto  
Class: InRequest IsStoredIn

#### **4.3.50 InRequestProcessHeader Class**

Parent Class:InDBAccess

Public:No

Distributed Object:No

Persistent Class:True

Purpose and Description:

Provides checkpoint storage of ingest request processing information associated with a given ingest request. In the event of request completion, this item is deleted.

#### **Attributes:**

**myDBAccess** - The pointer to the instance of the InDBAccess class used to connect to the database.

Data Type:InDBAccess\*

Privilege:Private

Default Value:

#### **Operations:**

**Delete** - Delete a requestID record.

Arguments:EcTInt RequestID, RWTime ProcessingEndDateTime

Return Type:EcUtStatus

Privilege:Public

**GetDataVolume** - Get the DataVolume for a given RequestID.

Arguments:const EcTInt requestID, EcTInt&dataVolume

Return Type:EcUtStatus

Privilege:Public

**GetGraphicalData** - Select RequestProcessHeader GUI graphical information for a given RequestID into a file.

Arguments:const EcTInt requestID, const RWCString fileName, EcTInt&rowCount

Return Type:EcUtStatus

Privilege:Public

**GetGraphicalData** - Select RequestProcessHeader GUI graphical information for all current requests into a file.

Arguments:const RWCString filename, EcTInt&rowCount

Return Type:EcUtStatus

Privilege:Public

**GetGraphicalData** - Select RequestProcessHeader GUI graphical information for a given Data Provider into a file.

Arguments:const RWCString externalDataProvider, const RWCString fileName, EcTInt&rowCount

Return Type:EcUtStatus

Privilege:Public

**GetInteractiveUserStatus** - Select request status information for a given user.

Arguments:RWCString ExternalDataProvider

Return Type:EcUtStatus

Privilege:Public

**GetReqDataByingestType** - Select RequestProcessHeader information for a given Ingest Type.

Arguments:const RWCString ingestType, const RWCString filename, EcTInt& rowCount

Return Type:EcUtStatus

Privilege:Public

**GetRequestData** - Select RequestProcessHeader information for a given RequestID into a file.

Arguments:const EcTInt requestID, const RWCString filename, EcTInt& rowcount

Return Type:EcUtStatus

Privilege:Public

**GetRequestData** - Select RequestProcessHeader information for a given RequestID into a file.

Arguments:const RWCString filename, EcTInt rowCount

Return Type:EcUtStatus

Privilege:Public

**GetRequestData** - Select RequestProcessHeader information for a given RequestID into a file.

Arguments:const RWCString externalDataProvider, const RWCString filename, EcTInt& rowCount

Return Type:EcUtStatus

Privilege:Public

**GetRequestMgrUR** - Get the ReqMgrUR for a given RequestID.

Arguments:const EcTInt requestID, RWCString&reqMgrUR

Return Type:EcUtStatus

Privilege:Public

**InRequestProcessHeader**

Arguments:InDBAccess\*

Return Type:Void

Privilege:Public

**Insert** - Insert a new record into the RequestProcessHeader table.

Arguments:EcTInt requestID, EcTInt reqPrio, EcTInt sequenceID, EcTInt sessionID, RWCString ingestType, RWCString extDataProv, RWTime expDateTime, EcTInt totDataVol, EcTInt totGranCnt, RWString mission, EcTInt totFileCnt, RWTime procStartTime, RWCString reqMgrUR

Return Type:Void

Privilege:Public

**RequestDataFetch** - Execute the SQL command to select data from the RequestProcessHeader table and fetch the results and store them in the input file.

Arguments:RWCString SQLCmd, RWCString filename, EcTInt&rowCount

Return Type:EcTuStatus

Privilege:Public

**RequestDataSelectCmd** - Create the select part of the SQL command to get the needed data from the RequestProcessHeader table.

Arguments:EcTVoid

Return Type:static RWCString

Privilege:Public

**UpdatePercentComplete**

Arguments:const EcTInt percentComplete, const EcTInt requestID

Return Type:EcUtStatus

Privilege:Public

**UpdatePriority** - Update the priority for a given Ingest request or Data Provider.

Arguments:const EcTInt requestPriority, const EcTInt requestID

Return Type:EcUtStatus

Privilege:Public

**UpdatePriority** - Update the priority for a given Ingest request or Data Provider.

Arguments:const EcTInt requestPriority, const RWCString externalDataProvider

Return Type:EcUtStatus

Privilege:Public

**UpdateState** - Update the Request State for a given Ingest request.

Arguments:const RWCString requestState, const EcTInt requestID

Return Type:EcUtStatus

Privilege:Public

**UpdateStateforStateChangeB** - Updates the state status for a given request.

Arguments:RWCString RequestState, int requestID

Return Type:EcUtStatus

Privilege:Public

**~InRequestProcessHeader**

Arguments:EcTVoid

Return Type:Void

Privilege:Public

#### **Associations:**

The InRequestProcessHeader class has associations with the following classes:

Class: InRequestController Access

Class: InSession Inserts

Class: InRequestSummaryData IsMovedTo

Class: InRequest isStoredIn

### **4.3.51 InRequestSummaryData Class**

Parent Class:InDBAccess

Public:No

Distributed Object:No

Persistent Class:True

Purpose and Description:

Provides long-term storage of summary data type statistics associated with a given data granule in a given ingest request.

#### **Attributes:**

**myDBAccess** - The pointer of the DBAccess object class.

Data Type:InDBAccess\*

Privilege:Private

Default Value:

**myDataGranuleID** - Numeric (ASCII) identifier of a data granule within an ingest request. Determined incrementally for each data granule in an ingest request.

Data Type:int

Privilege:Private

Default Value:

**myDataGranuleVolume** - Total data volume to be ingested for a data granule in an ingest request. The total data volume for the data granule is determined by summing the data volumes for the files comprising the data granule.

Data Type:int

Privilege:Private

Default Value:

**myDataType** - Data type identifier for the data granule. Selected from a list of valid data type identifiers maintained by the Data Server.

Data Type:RWCString

Privilege:Private

Default Value:

**myFinalStatus** - Final error status for the ingest processing of a data granule.

Data Type:int

Privilege:Private

Default Value:

**myRequestID** - Identifier of the InRequest\_S object to which this entry corresponds. This is a primary key.

Data Type:int

Privilege:Private

Default Value:

**myRetryCount** - Indicates the number of retries Ingest should attempt before returning an error.

Data Type:int

Privilege:Private

Default Value:

**myTimeToArchive** - Time from submit of archive request to Data Server to receipt of completion status (success or fail).

Data Type:int

Privilege:Private

Default Value:

**myTimeToPreprocess** - Time from start of preprocessing of granule to time of completion (success or fail) of preprocessing.

Data Type:int

Privilege:Private

Default Value:

**myTimeToXfer** - Time from start of transfer for 1st file in granule to time of receipt of status (success or failure) for last file in granule.

Data Type:int

Privilege:Private

Default Value:

**myTotalFileCount** - Total number of files in the granules.

Data Type:int

Privilege:Private

Default Value:

## Operations:

**GetSummaryData** - Select Request Summary Data information for a given request into a file.

Arguments:const EcTInt requestID, const RWCString filename, EcTInt& rowCount

Return Type:EcUtStatus

Privilege:Public

### **InRequestSummaryData**

Arguments:InDBAccess\* DBAccess

Return Type:Void

Privilege:Public

### **~InRequestSummaryData**

Arguments:void

Return Type:Void

Privilege:Public

## Associations:

The InRequestSummaryData class has associations with the following classes:

Class: InHistoryLog Accesses

Class: InRequestProcessData IsMovedTo

Class: InRequestProcessHeader IsMovedTo

Class: InRequestProcessData IsMovedto

### 4.3.52 InRequestSummaryHeader Class

Parent Class:InDBAccess

Public:No

Distributed Object:No

Persistent Class:True

Purpose and Description:

Provides long-term storage of summary request-level statistics associated with a given ingest request.

#### Attributes:

**myDBAccess** - The pointer to the instance of the InDBAccess class used to connect to the database.

Data Type:InDBAccess\*

Privilege:Private

Default Value:

#### Operations:

**GetStatistics** - Based on search criteria from GUI, query summary statistical data (average, minimum, maximum request totals or request times).

Arguments:const InTHLSearchCriteria, const RWCStringfilename, int rowCount

Return Type:EcUtStatus

Privilege:Public

**GetSummaryHeader** - Select Request Summary Header information for a given request into a file.

Arguments:const InTHLSearchCriteria searchCriteria, const RWCString filename, EcTInt rowCount

#### **InRequestSummaryHeader**

Arguments:InDBAccess\* DBAccess

Return Type:Void

Privilege:Public

#### **~InRequestSummaryHeader**

Arguments:void

Return Type:Void

Privilege:Public

**Associations:**

The InRequestSummaryHeader class has associations with the following classes:

Class: InHistoryLog Accesses

**4.3.53 InSDMetadata Class**

Parent Class: InMetadata

Public: No

Distributed Object: No

Purpose and Description:

This class provides services to preprocess data which is in self-descriptive format other than HDF.

**Attributes:**

All Attributes inherited from parent class

**Operations:**

**InSDMetadata** - This is the constructor service.

Arguments:

**Preprocess** - This operation will extract values from input files with a self-descriptive format (other than HDF). It will access the correct MCFs and interact with InMetadataTool Class to produce a metadata file which is acceptable to the Data Server Subsystem.

Arguments:

**~InSDMetadata**

Arguments:

Return Type: Void

Privilege: Public

**Associations:**

The InSDMetadata class has associations with the following classes:

None

#### 4.3.54 InScienceData Class

Parent Class:Not Applicable  
Public:No  
Distributed Object:No  
Purpose and Description:  
This is an abstract class

##### Attributes:

**myInputScienceFile** - The file name of the original science data.

Data Type:InFile\*

Privilege:Private

Default Value:

**myOutputScienceFile** - The file name of the new science data generated by the ingest preprocessing.

Data Type:InFile\*

Privilege:Private

Default Value:

##### Operations:

**Preprocess** - This is an abstract operation.

Arguments:

Return Type:Void

Privilege:Public

This is an abstract operation

##### Associations:

The InScienceData class has associations with the following classes:

InDataPreprocessTask (Aggregation)

#### 4.3.55 InServer Class

Parent Class:EcPFManagedServer

Public:No

Distributed Object:Yes

Persistent Class:

Purpose and Description:

Provides a single point of entry to the Ingest system for all ingest interfaces. The object class manages ingest sessions.

#### **Attributes:**

**mySessionCount** - The total number of sessions running under the Ingest Server.

Data Type:int

Privilege:Private

Default Value:

#### **Operations:**

**StartServer** - Starts up the Ingest Server.

Arguments:void

Return Type:int

Privilege:Public

#### **Associations:**

The InServer class has associations with the following classes:

Class: InSessionInfo IsManagedBy

Class: InSession Manages - All instance of InSession object is managed by one instance of InServer object.

#### **4.3.56 InServerExtRPC\_C Class**

Parent Class:Not Applicable

Public:Yes

Distributed Object:Yes

Purpose and Description:

This is the client/proxy implementation that defines the RPC for initiating an Ingest Session.

**Attributes:**

None

**Operations:**

**CreateSession** - This is a RPC that initiates a new Ingest Session.

Arguments:handle\_t InServerBH, char \*GatewayStringBH, error\_status\_t  
\*CreateSessStatus

Return Type:int

Privilege:Public

**Associations:**

The InServerExtRPC\_C class has associations with the following classes:

Class: CsGateWay IsInvokedBy - The Gateway object interfaces with the InServerExtRPC to initiate a new Ingest Session through the Ingest Server.

InServer (Aggregation)

**4.3.57 InServerExtRPC\_S Class**

Parent Class:Not Applicable

Public:No

Distributed Object:Yes

Purpose and Description:

This is the server implementation that defines the services for creating a new session.

**Attributes:**

None

**Operations:**

**CreateSession** - Creates a new session for a given client upon receipt of an Authentication Request.

Arguments:handle\_t InServerBH, char \*GatewayStringBH, error\_status\_t

\*CreateSessStatus  
Return Type:Void  
Privilege:Public

#### **Associations:**

The InServerExtRPC\_S class has associations with the following classes:  
InServer (Aggregation)

### **4.3.58 InServerIntRPC\_C Class**

Parent Class:Not Applicable  
Public:Yes  
Distributed Object:Yes  
Purpose and Description:  
This is the client/proxy implementation for the InServer object class. The provided services are to be used by the InSession object class.

#### **Attributes:**

None

#### **Operations:**

**DeleteSession** - Deletes the specified session from the InServer's Session List.  
Arguments:handle\_t InServerBH, int SessionId, error\_status\_t \*DelSessStatus  
Return Type:Void  
Privilege:Public

#### **Associations:**

The InServerIntRPC\_C class has associations with the following classes:  
Class: InSession IsInvokedBy - InSession interfaces with InServerIntRPC\_C to delete itself from the InServer's session list.  
InServer (Aggregation)

### **4.3.59 InServerIntRPC\_S Class**

Parent Class:Not Applicable

Public:No

Distributed Object:Yes

Purpose and Description:

This is the server implementation (factory) for the InServer object class. The provides services are to be used by the InSession object class.

#### **Attributes:**

None

#### **Operations:**

**DeleteSession** - Deletes the specified session from the InServer's Session List.

Arguments:handle\_t InServerBH, int SessionId, error\_status\_t \*DelSessStatus

Return Type:Void

Privilege:Public

#### **Associations:**

The InServerIntRPC\_S class has associations with the following classes:

InServer (Aggregation)

### **4.3.60 InSession Class**

Parent Class:EcPFManagedServer

Public:No

Distributed Object:Yes

Persistent Class:

Purpose and Description:

This is the super object class for specialization object classes that handle specific external interfaces. In general, the object class manages the hand-shaking protocol with the ingest service requestor. It verifies that the requestor has privilege to perform the data ingest service. The InSession instantiates the InRequest and adds to the InRequestList to be processed. In addition, the InSession allows cancellation, suspension, and resumption of the Ingest Request processing running under the session. Suspension and resumption are post Release A functions.

## Attributes:

**myClientId** - Session's client identifier.

Data Type:char \*

Privilege:Private

Default Value:

**mySessionGWBH** - Session's binding handle with the Gateway.

Data Type:char \*

Privilege:Private

Default Value:

**mySessionId** - The information that uniquely identifies the session.

Data Type:int

Privilege:Private

Default Value:

## Operations:

**InitSessServer** - Starts up the session. The operation is invoked by the Ingest Server upon receipt of an Create Session request from Gateway.

Arguments:char \*GatewayBH

Return Type:int

Privilege:Public

**ProcessRequest** - Once DAN is received from the Client, this operation instantiates a new Request and adds the request to the Ingest Request List, and sends DAA (DAN Acknowledgement) to the Client.

Arguments:void

Return Type:int

Privilege:Public

**ResumeSession** - Resumes the session. All ingest processing running under the session will be resumed. This is a post Release A service.

Arguments:void

Return Type:int

Privilege:Public

**SuspendSession** - Suspends the session. All ingest processing running under the session will be suspended. This is a post Release A service.

Arguments:void

Return Type:int

Privilege:Public

**TerminateSession** - Terminates the session. All ingest processing running under the session will be terminated.

Arguments:void

Return Type:int

Privilege:Public

### **Associations:**

The InSession class has associations with the following classes:

Class: InRequestProcessHeader Inserts

Class: InRequestManager\_C Invokes

Class: InServerIntRPC\_C IsInvokedBy - InSession interfaces with InServerIntRPC\_C to delete itself from the InServer's session list.

Class: InSessionEcsRPC\_C IsInvokedBy

Class: InServer Manages - All instance of InSession object is managed by one instance of InServer object.

Class: InMessage Receives/Sends - The InSession object interfaces with the InMessage object to access data messages that are interchanged between Ingest and the external Client.

Class: InRequestManager\_C ReceivesRequestFrom

Class: InRequestManager\_C SendsRequestTo

### **4.3.61 InSessionEcsRPC\_C Class**

Parent Class:Not Applicable

Public:Yes

Distributed Object:Yes

Purpose and Description:

This is the client/proxy implementation that defines services for sending outgoing data messages from the ECS Ingest.

### **Attributes:**

None

### **Operations:**

**ecsDDN** - The RPC is invoked by ECS Ingest to send the Data Delivery Notice (DDN) data message to the external client.

Arguments:handle\_t GatewayBH, char \*DDN, error\_status\_t ecsDDNstatus

Return Type:Void  
Privilege:Public

**Associations:**

The InSessionEcsRPC\_C class has associations with the following classes:

Class: InSession IsInvokedBy  
CsGateWay (Aggregation)

#### **4.3.62 InSessionEcsRPC\_S Class**

Parent Class:Not Applicable

Public:No

Distributed Object:Yes

Purpose and Description:

This is the server implementation that defines services for sending outgoing data messages from ECS Ingest to external client.

**Attributes:**

None

**Operations:**

**ecsDDN** - The RPC is invoked by ECS Ingest to send DDN (Data Delivery Notice) data message to the external client.

Arguments:handle\_t GatewayBH, char \*DDNmsg, error\_status\_t ecsDDNstatus

Return Type:Void

Privilege:Public

**Associations:**

The InSessionEcsRPC\_S class has associations with the following classes:

CsGateWay (Aggregation)

### 4.3.63 InSessionExtRPC\_C Class

Parent Class:Not Applicable

Public:Yes

Distributed Object:Yes

Purpose and Description:

This is the client/proxy implementation that defines the RPC (Remote Procedure Call) for delivering data message from the external Client to ECS/Ingest.

#### Attributes:

None

#### Operations:

**extDAN** - This is the RPC that delivers the Data Availability Notice (DAN) data message from the external client to ECS Ingest.

Arguments:handle\_t InSessBH, char \*DANmsg, char \*\*DAAMsg, error\_status\_t \*extDANstatus

Return Type:Void

Privilege:Public

**extDDA** - The is the RPC that delivers the Data Delivery Ack (DDA) from the external client to ECS Ingest.

Arguments:handle\_t InSessBH, char \*DDAMsg, error\_status\_t \*status

Return Type:Void

Privilege:Public

#### Associations:

The InSessionExtRPC\_C class has associations with the following classes:

Class: CsGateWay Invokes - The Gateway object interfaces with the InSessionExtRPC object to deliver the DAN and DDA data messages received from the external Client to Ingest Session.

InSession (Aggregation)

### 4.3.64 InSessionExtRPC\_S Class

Parent Class:Not Applicable

Public:No

Distributed Object:Yes

Purpose and Description:

This is the server implementation that defines services for sending data messages from the external client to ECS Ingest.

**Attributes:**

None

**Operations:**

**extDAN** - This is the RPC that delivers the Data Availability Notice (DAN) data message to ECS Ingest.

Arguments:handle\_t InSessBH, char \*DANmsg, char \*\*DAAMsg, error\_Status\_t \*extDANstatus

Return Type:Void

Privilege:Public

**extDDA** - This is the RPC that delivers the Data Delivery Ack (DDA) from the external client to ECS Ingest.

Arguments:handle\_t InSessBH, char \*DDAMsg, error\_status\_t \*extDDAstatus

Return Type:Void

Privilege:Public

**Associations:**

The InSessionExtRPC\_S class has associations with the following classes:

InSession (Aggregation)

**4.3.65 InSessionInfo Class**

Parent Class:InDBAccess

Public:No

Distributed Object:No

Persistent Class:True

Purpose and Description:

Keeps track of all the sessions running under the Ingest Server.

**Attributes:**

**ClientID** - The identifier of the external client.

Data Type:char \*

Privilege:Private

Default Value:

Constraints:

Non Persistent Flag:False

**SessionID** - The identifier of the Ingest Session.

Data Type:int

Privilege:Private

Default Value:

Constraints:

Non Persistent Flag:False

**Operations:**

**AddSession** - Adds a session to the Ingest Server's session list.

Arguments:char \*ClientID, int SessionID

Return Type:int

Privilege:Public

**DeleteSession** - Deletes a session from the Ingest Server's session list.

Arguments:int SessionID

Return Type:int

Privilege:Public

**ListSessions** - Lists all sessions in the Ingest Server's session list.

Arguments:void

Return Type:int

Privilege:Public

**SearchSession** - Searches for a session in the Ingest Server's session list based on the external client identifier.

Arguments:char \*ClientID

Return Type:int

Privilege:Public

**SearchSession** - Searches for a session in the Ingest Server's session list based on the session identifier.

Arguments:int SessionID

Return Type:int

Privilege:Public

#### **Associations:**

The InSessionInfo class has associations with the following classes:

Class: InServer IsManagedBy

### **4.3.66 InSessionIntRPC\_C Class**

Parent Class:Not Applicable

Public:Yes

Distributed Object:Yes

Purpose and Description:

This is the client/proxy implementation for exporting the data messages to the InSession object class.

#### **Attributes:**

None

#### **Operations:**

**IntDDN** - This is the RPC that exports the Data Delivery Notice (DDN) data message to the InSession object class.

Arguments:handle\_t InSessBH, char \*DDN, error\_status\_t IntDDNstatus

Return Type:Void

Privilege:Public

**Associations:**

The InSessionIntRPC\_C class has associations with the following classes:  
InSession (Aggregation)

**4.3.67 InSessionIntRPC\_S Class**

Parent Class:Not Applicable

Public:No

Distributed Object:Yes

Purpose and Description:

This is the server implementation for exporting data messages to the InSession object class.

**Attributes:**

None

**Operations:**

**InDDN** - This is the RPC that exports the Data Delivery Notice (DDN) data message to the InSession object class.

Arguments:handle\_t InSessBH, char \*DDN, error\_status\_t InDDNstatus

Return Type:Void

Privilege:Public

**Associations:**

The InSessionIntRPC\_S class has associations with the following classes:  
InSession (Aggregation)

**4.3.68 InShortDAA Class**

Parent Class:InMessage

Public:No

Distributed Object:No

Purpose and Description:

This object class populates the short DAA (DAN Acknowledgement) data message to be sent to the external Client after the receipt of the DAN.

**Attributes:**

**myShortDAA** - This is the short DAA (DAN Acknowledgement) data message.

**Operations:**

**FillDAA** - This function will package the short DAA for the DAN Acknowledgement.  
Arguments: int DAASStatus, int DANSeqNo

**Associations:**

The InShortDAA class has associations with the following classes:  
None

#### **4.3.69 InShortDDN Class**

Parent Class: InMessage

Public: No

Distributed Object: No

Purpose and Description:

This object class populates the short DDN (Data Delivery Notice) data message to be sent to the external Client after the data is archived.

**Attributes:**

**myShortDDN** - Short Data Delivery Notice (DDN) data message.

Data Type: Short DDN msg

Privilege: Private

Default Value:

**Operations:**

**FillDDN** - Populates the short DDN data message with the given status information.

Arguments:int DDNStatus, int DANSeqNo

Return Type:Void

Privilege:Public

#### **Associations:**

The InShortDDN class has associations with the following classes:

Class: InRequest builds

### **4.3.70 InSnowIceData Class**

Parent Class:InScienceData

#### **Attributes:**

All Attributes inherited from parent class

#### **Operations:**

**InSnowIceData** - This is the constructor service.

Arguments:

Return Type:Void

Privilege:Public

**Preprocess** - This service provides the functionality to preprocess Snow/Ice Data.

Arguments:

Return Type:EcUtStatus

Privilege:Public

**~InSnowIceData** - This is the destructor service.

Arguments:

Return Type:Void

Privilege:Public

#### **Associations:**

The InSnowIceData class has associations with the following classes:

None

#### 4.3.71 InSourceMCF Class

Parent Class:InDBAccess

Public:No

Distributed Object:No

Persistent Class:True

Purpose and Description:

This class retains configuration information on source input files (i.e., source parameter name, parameter location). This class provides services to retrieve, delete, and add configuration information for a specific source metadata configuration.

##### Attributes:

**myCmd** - This attribute specifies the Sybase command pointer.

Data Type:CS\_COMMAND

Privilege:Private

Default Value:

**myCtr** - This attribute is the local counter.

Data Type:EcTInt

Privilege:Private

Default Value:

**myDataType** - This attribute specifies the data type (e.g., CER00, LIS00) associated with the source metadata configuration file.

Constraints:

Non Persistent Flag:False

**myFileType** - This attribute specifies the file type (e.g. metadata, science) associated with the source metadata configuration file.

Constraints:

Non Persistent Flag:False

**myFilename\_vector** - This attribute defines the ordered vector which contains a list of source mcf rows.

Data Type:RWTPtrOrderedVector <ParInfo>

Privilege:Private

Default Value:

## **myMCFTable**

**myNumCols** - This attribute specifies the number of columns in the result set.

Data Type:CS\_INT

Privilege:Private

Default Value:

**myParInfo** - This attribute references the structure which stores parameter information.

Data Type:ParInfo\*

Privilege:Private

Default Value:

**myRetCode** - This attribute specifies the Sybase return status.

Data Type:CS\_RETCODE

Privilege:Private

Default Value:

**mySQLCmd** - This attribute specifies the SQL command pointer.

Data Type:RWCString\*

Privilege:Private

Default Value:

**mySourceMCF** - This attribute references the appropriate SourceMCF.

Data Type:RWCString\*

Privilege:Private

Default Value:

## **Operations:**

**GetNextPar** - This service provides a structure representing parameter information on the next applicable parameter of the SourceMCF.

Arguments:ParInfo\*\* ParHolder

Return Type:EcUtStatus

Privilege:Public

**GetSourceMCF** - This service downloads the applicable Source MCF information from the Ingest database.

Arguments:RWCString SourceMCF

Return Type:EcUtStatus

Privilege:Public

**InSourceMCF** - This is the default constructor service.

Arguments:void

**InSourceMCF** - This is the constructor service.

Arguments:RWCString SourceMCF

Return Type:void

Privilege:Public

**~InSourceMCF** - This is the destructor service.

Arguments:void

Return Type:void

Privilege:Public

### **Associations:**

The InSourceMCF class has associations with the following classes:

Class: InMetadata defines - The InSourceMCF class defines the InMetadata class by providing format informatin on input metadata files.

## **4.3.72 InTOMSData Class**

Parent Class:InScienceData

### **Attributes:**

All Attributes inherited from parent class

### **Operations:**

**InTOMSData** - This is the constructor service.

Arguments:

Return Type:Void

Privilege:Public

**Preprocess** - This service provides the functionality to preprocess TOMS data.

Arguments:

Return Type:EcUtStatus

Privilege:Public

**~InTOMSData** - This is the destructor service.

Arguments:  
Return Type:Void  
Privilege:Public

#### Associations:

The InTOMSDData class has associations with the following classes:  
None

## 4.4 Ingest CSCI Dynamic Model

Information is provided for the critical Ingest Subsystem scenarios. The scenarios are each in two parts--the set of scenario steps and an event trace diagram. The set of scenario steps provides a text description of the interactions between object classes in a scenario. The event trace diagram pictorially describes the interaction between object classes and external interfaces participating in a scenario.

### 4.4.1 Automated Network Ingest (Get) Scenario

ECS performs automated network ingest upon receipt of a Data Availability Notice (DAN) stimulus. The Landsat-7 LPS, SDPF, TSDIS, and SCF interfaces are candidates for use of this protocol. In this scenario, ECS will perform the data transfer (get) from the external location to the ECS system.

If the external data provider is not a DCE client, there will an ECS Gateway that will translate the TCP/IP socket service class received from the external data provider to the corresponding Remote Procedure Call (RPC) provided by the ECS Ingest. In the scenario diagram, the "External Data Provider Process" would be the Gateway if the external data provider does not have DCE.

The following list describes the Automated Network Ingest (Get) scenario using object classes. Figure 4.4-1 is the corresponding event trace diagram. The numbers in the following list refer to the steps in the diagram.

**Table 4.4-1. Automated Network Ingest Scenario (Get) Event Trace Diagram (1 of 3)**

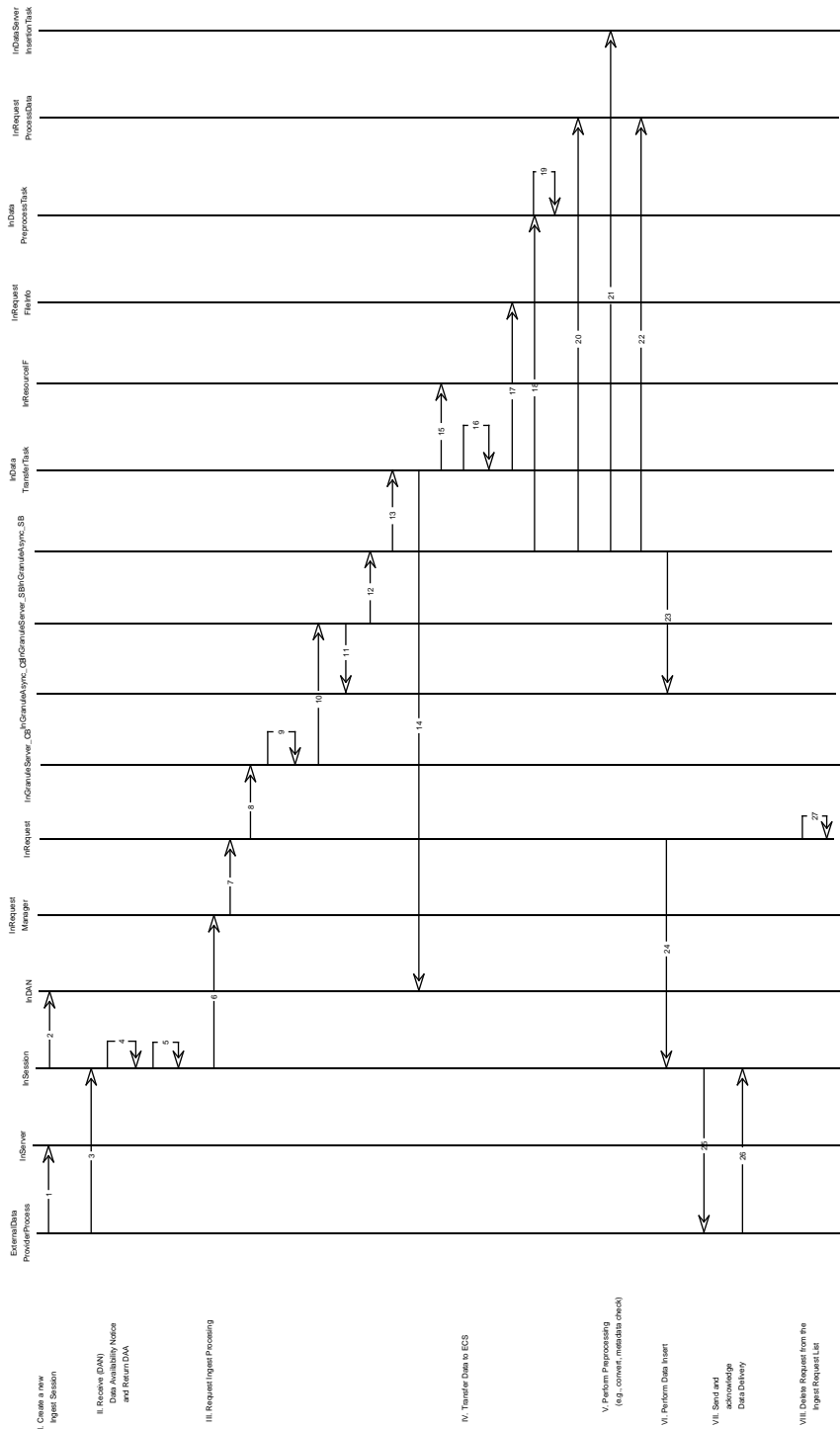
Step	Service	Description
1	CreateSession()	The Ingest server instantiates a new Ingest Session and setup the connection between Ingest Session and the External Data Provider process
2	extDAN()	External Data Provider process sends a DAN (Data Availability Notice) message to Ingest system. The DAN is verified and a DAA (DAN Ack) is returned
3	WriteEvent()	Log the receipt of DAN

**Table 4.4-1. Automated Network Ingest Scenario (Get) Event Trace Diagram (2 of 3)**

Step	Service	Description
4	GetRequestId() (Next Available)	Assign an unique ID for the Ingest Request
5	Checkpoint Request () InRequestProcess Data () InRequestFileInfo ()	Checkpoint all DAN Granules and file information to database Partition DAN into granules (File Groups) for Remote Granule Processing
6	ProcessRequest()	Request to perform the Automated Network Ingest Request
7	Constructor	Instantiate Request object
8	Construct	In Granule Server_C object. Determine UR of Server to Process Request Build Process Granule Message
9	Build MSG()	Build Process Granule Message
10	Process Granule ()	Send Process Granule Message to appropriate InGranuleRequestServer_S
11	Receive MSG()	Receive Process Granule Message. Construct InGranuleAsync_C object
12	Constructor	Construct InGranule Async_S object
13	Constructor	Instantiate InDataTransfer object
14	Constructor InDataTypeList()	Construct DAN granule object and then populate list of data types to be processed
15	AllocateResource()	Request allocation of an available staging device
16	Transfer() Data	Transfer all files in the granule from external location to the resource object location in ECS Ingest System
17	Update Status	Update Granule Checkpoint information in database (Submitted for archive)
18	Constructor	Instantiate InDataPreProcessTask object
19	PreprocessData()	Performs appropriate data preprocessing (e.g., metadata check, conversion)
20	Update Status	Update Granule Checkpoint information in database (Preprocessed)
21	SendInsert()	Send Insert Request to the appropriate DataServer
		Refer to DID 313, Internal Interfaces, Section 4.2.12, Insert_Data_SP12 for data insertion details
22	Update Status	Update Granule Checkpoint information in database (Archived)
23	Send MSG()	Send Granule complete message

**Table 4.4-1. Automated Network Ingest Scenario (Get) Event Trace Diagram (3 of 3)**

Step	Service	Description
		Repeat Steps 8-23 for each DataType
24	IntDDN	Send DDN (Data Delivery Notice) to Session
25	ecsDDN	Send DDN (Data Delivery Notice) to the External Data Provider process providing Data Archive status
26	extDDA	The External Data Provider process returns DDA (Data Delivery Ack)
27	DeleteRequest()	Removes the Ingest Request from the Ingest Checkpoint database



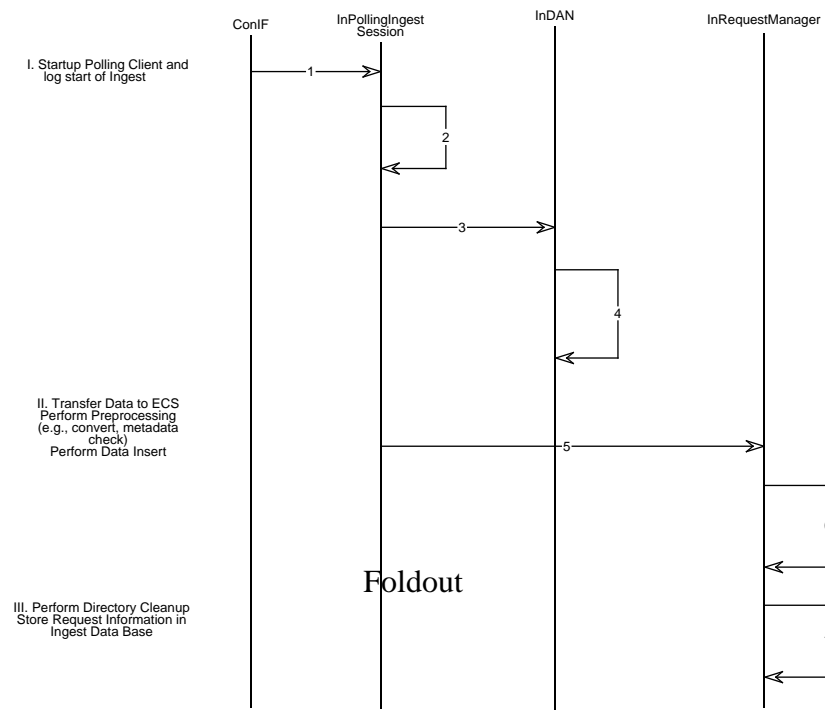
**Figure 4.4-1. In\_Automated\_Network\_Ingest\_Get\_Event\_Trace\_Diagram**

#### 4.4.2 Polling Ingest (Files) Scenario

The Polling Ingest (Files) scenario describes the mechanism by which the Ingest Subsystem acquires data from data centers which may not support an interprocess communication interface with ECS. The ECS/NESDIS and ECS/DAO interfaces are candidates for use of this scenario. The following list describes the Polling Ingest (Files) Scenario. Figure 4.4-2 is the corresponding event trace diagram.

**Table 4.4-2. Polling Ingest (Files) Event Trace**

Step	Service	Description
1	Constructor	Instantiate polling session (start timer) to detect new files from the specified external directory
2	ProcessRequest()	Initiate Ingest processing
3	constructor	Instantiate InDAN object class
4	GenerateDAN()	Generate a DAN file with file information retrieved from the specified directory location
5	(InRequestManager())	See Table 4.4-1, Automated Network Ingest, Steps 4-23 for request processing steps
6	CleanupDirectory()	Perform appropriate directory cleanup by means of moving the completed files to another directory
7	Delete()	Log Completion of Request Delete Checkpointed Information
		Repeat steps 2-7 for ingesting next directory



**Figure 4.4-2. In\_Polling\_Files\_Ingest\_Event\_Trace Dynamic Model**

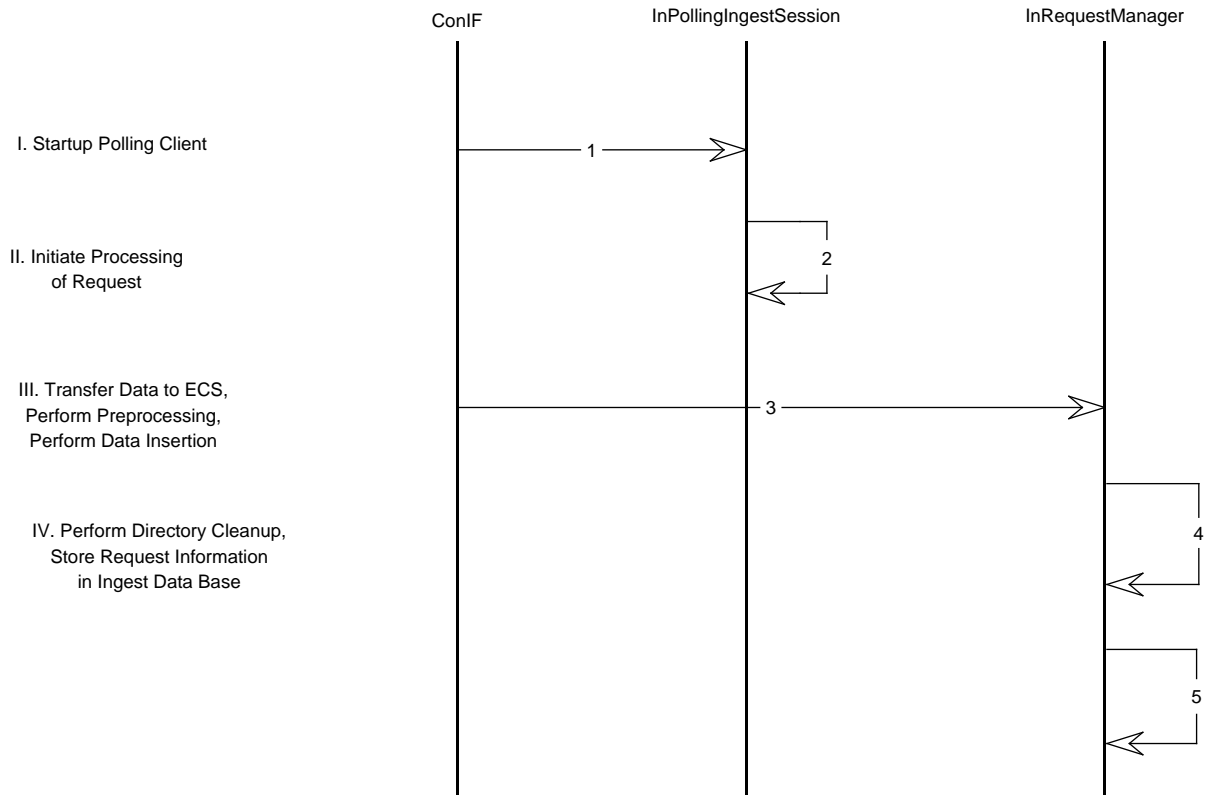
#### 4.4.3 Polling Ingest (Delivery Record) Scenario

The Polling Ingest (Delivery Record) scenario describes the mechanism by which the Ingest Subsystem acquires data from External I/Fs which control the initiation of data transfer. The EDOS, LaRC DAAC, GSFC DAAC, ORNL DAAC, ACRIM, and SAGE-III interfaces are candidates for use of the polling with delivery record scenario.

The following list describes the Polling Ingest (Delivery Record) Scenario. Figure 4.4-3 is the corresponding event trace diagram.

**Table 4.4-3. Polling Ingest (Delivery Record) Event Trace Diagram**

Step	Service	Description
1	Constructor	Instantiate polling session (start timer) to detect the Delivery Record file from the specified ECS directory
2	ProcessRequest() (InRequestManager())	Initiate Processing of request See Table 4.4-1, Automated Network Ingest, Steps 4-23 for Request processing steps
3	CleanupDirectory()	Perform appropriate Directory Cleanup by means of moving the completed files to another directory
4	Delete()	Store Request Summary information in Ingest Database and Delete Request Checkpoint Data



**Figure 4.4-3. In\_Polling\_Delivery\_Record\_Ingest\_Event Trace Diagram**

#### 4.4.4 Interactive Ingest Scenario

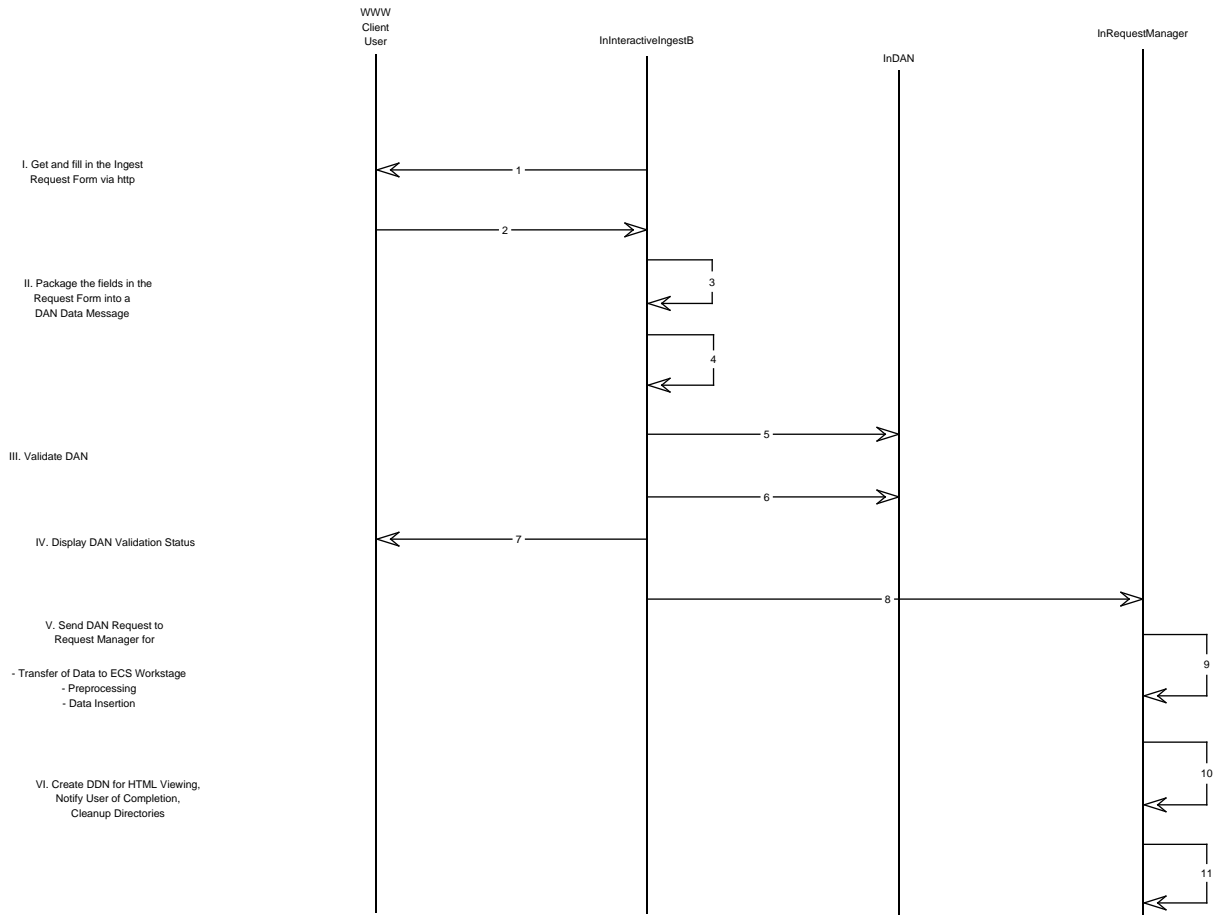
The ECS provides the ECS users with the capability to perform interactive Ingest via the HTML Forms Interface. An interactive user will connect via World Wide Web (WWW) to the ECS Interactive Ingest Interface. The user, via HTML Forms, will submit a request to the ECS system for data ingest.

The Interactive Ingest Interface provides four methods for the user/operator to define an ingest request. See Section 4.5.7 Interactive Ingest Interface CSC for details. Note: The phrase data set is used interchangeably with the word granule throughout these sections. The following list describes one of these methods. In this Interactive Ingest scenario, the user has chosen to specify

the directory (or directories) where a data set (or data sets) resides. Figure 4.4-4 is the corresponding event trace diagram. The numbers in the following list refer to the steps in the diagram.

**Table 4.4-4. Interactive Ingest Event Trace**

Step	Service	Description
1	N/A	User retrieves a Interactive Ingest Request Form via the WWW client
2	N/A	The Interactive Ingest Request Form is filled by the user and is submitted to the Ingest Form Script via HTTP daemon. (The HTTP daemon will parse the fields from the Form and invoke Ingest Form Script) . The user has selected the data type directory specification method of Ingest. The files for each data set to be ingested have been pre-transferred by the user to ECS configured and controlled directories. External users are required to use KFTP to transfer files into ECS. Local users (operators) can copy files to configured directories. Write access to configured directories is operating system controlled
3	N/A	The Ingest Form Script processes the Interactive Ingest Request Form
4	ProcessFiles()	The Ingest Form Script will, via DataType/FileType Form, specify the data type of the data set to be ingested. The Ingest Form Script will validate the data types then process all files in the ECS configured directory for that data type. All file names will be displayed back to user. The user is required to type in a file type for each file
	N/A	The user can choose to repeat Step 4 for additional data sets. Each data set must reside in a separate directory
5	GenerateDAN	Generate a DAN file from the information retrieved from the specified directories and the DataType/FileType Form
6	extDAN()	The DAN is verified
7	N/A	The Ingest Form Script displays the status to the WWW client
8	InRequestManager	See Table 4.4-1, Automated Network Ingest, Steps 4-23 for request processing steps
9	Write DDN()	When the ECS Ingest System completes the Data Ingest, it writes Completion Notification Information for HTML viewing
10	Delete()	Store request summary information in the Ingest Database Delete request checkpoint data. If the Ingest was successful, Ingest software will delete user files from the ECS controlled data type directories. If the ingest failed, the Ingest software will move the files to a user accessible error directory. The user can transmit corrected versions of the files. Then, via the Interactive Ingest Interface, request the Ingest of the corrected dataset
11	NA	The user will be notified via E-Mail of request completion. E-Mail will include detail data set(s) file status information. The user can also view last completed Data Deliver Notice (DDN) via the HTML status form.
	NA	See Section 4.4.8, User Ingest Status Monitoring for details



**Figure 4.4-4. In\_Interactive\_Ingest\_Event\_Trace\_Diagram\_Dynamic Model**

#### 4.4.5 Hard Media Ingest Scenario

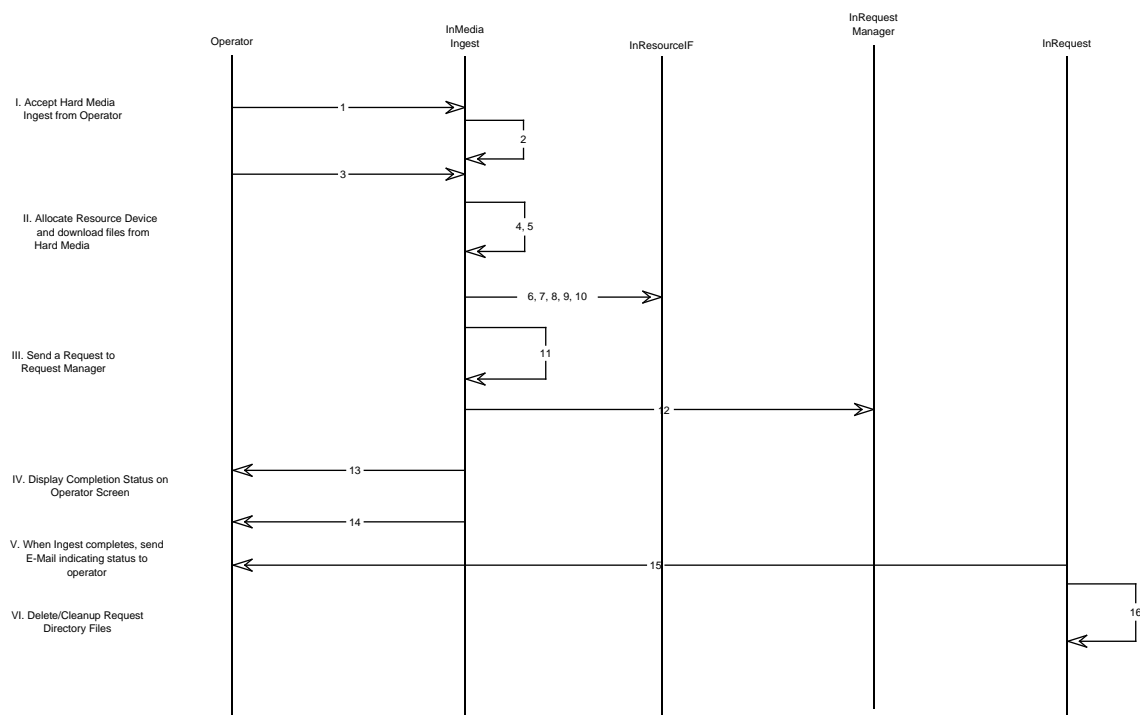
The ECS system provides operations personnel with the capability to perform hard media (e.g., 8mm tape) ingest via the GUI Interface. A Media Ingest Session will be configured on the Operator's GUI Interface to accept the request from the Operator via the GUI Interface and submit the request to the ECS system for hard media ingest. The ASTER GDS interface and backup

interfaces for other external data providers (e.g., SDPF and EDOS) are candidates for use of the hard media ingest scenario.

The following list describes the Hard Media Ingest scenario using object classes. Figure 4.4-5 is the corresponding event trace diagram. The numbers in the following list refer to steps in the diagram:

**Table 4.4-5. Hard Media Ingest Event Trace**

Step	Service	Description
1	Constructor	Operator selects the Media Ingest option from the GUI interface and instantiates a media interface client to read inputs from the operator
2	CheckPrivilege()	The Media Ingest Session verifies Operator's privilege
3	ReceiveMsg()	Operator enters information needed for media ingest and the information is read in
4	ProcessRequest()	Request to perform the media ingest request
5	DownloadFiles()	Request to download files from the media to disk
6	AllocateResource()	Allocate an available staging device
7	AllocateResource()	Allocate an available peripheral device
8	CopyFile()	Copy files from the media to disk working area
9		
10	DeallocateResource()	Deallocate the peripheral device
11	GetRequestID (Next Available(), InRequestProcess Header)	Assign Unique Request ID and Checkpoint Request information into Ingest Database after reading Media Delivery Record supplied with media
12	extDAN()	A DAN (Data Availability Notice) message is sent to the Ingest system. The DAN is verified.
13	SendMsg()	The Media Ingest Session displays the status to Operator screen
14	(InRequestManager())	See Table 4.4-1, Automated Network Ingest, Steps 5-23, for Request Processing Steps
15	ecsDDN()	When the Ingest system completes the data ingest, it sends a DDN (Data Delivery Notice) to Request via E-Mail. The Ingest Operator is notified of Completion Status on the Operator Screen
16	Delete() (InResourceIF, InRequestProcessHeader)	Cleanup Temporary Directories, Release Staging Resources, Store Request Summary information in Ingest Database and Delete Request Checkpoint Data



**Figure 4.4-5. In\_Hard\_Media\_Ingest\_Event\_Trace\_Diagram Dynamic Model**

#### 4.4.6 Ingest History Log Viewing Scenario

The ECS system provides operations personnel with the capability to view the IngestRequestSummaryLog, a log that contains the results of all the past ECS ingest requests. The operations personnel have the capability to specify the search criteria (e.g., time range), the provider ID, data set name, and final request status of the IngestRequestSummaryLog for log display. A History screen will be configured on the Operator's GUI Interface to accept the log monitor request and the criteria specification from the operations personnel and invoke the appropriate service (InRequestSummaryHeader) to get and display the Ingest History Log information to the operations personnel's GUI screen.

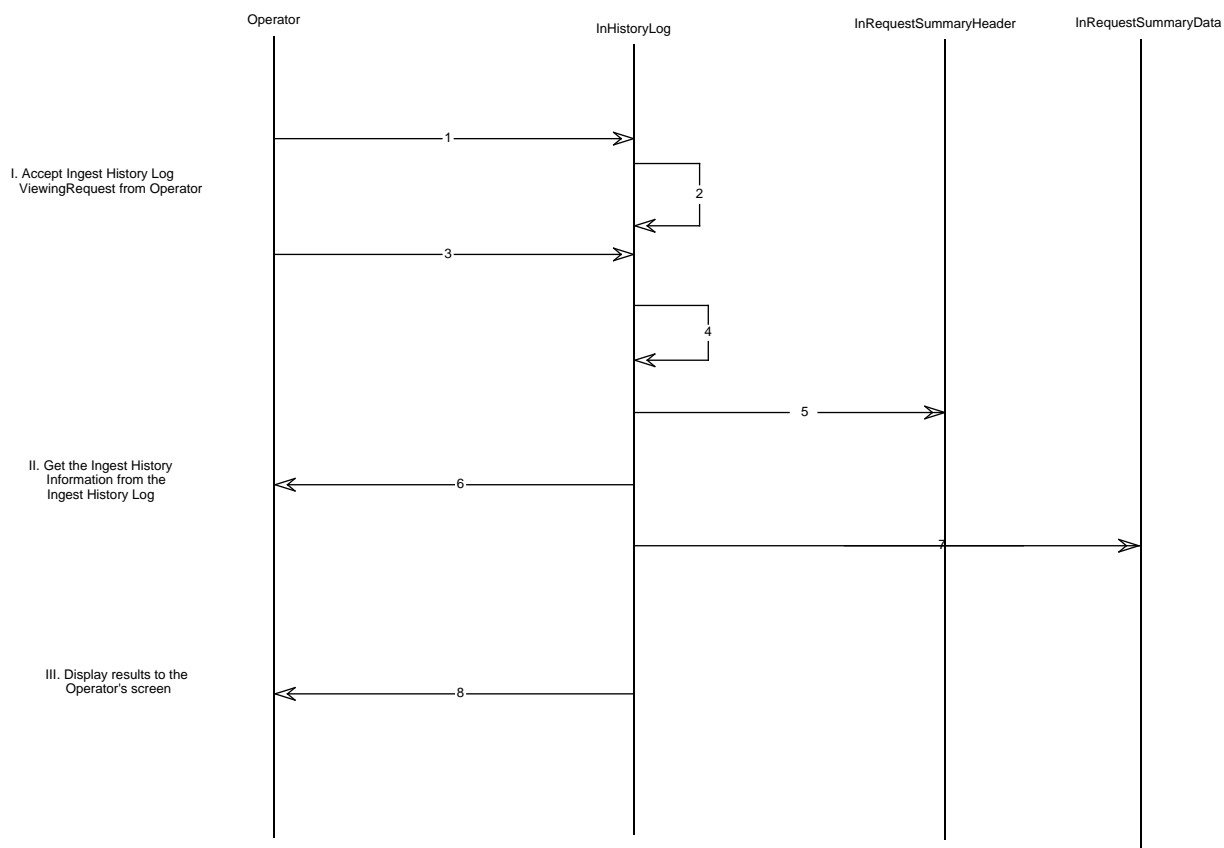
The following list describes the Ingest History Log Viewing scenario using object classes. Figure 4.4-6 is the corresponding event trace diagram. The numbers in the following list refer to the steps in the diagram.

**Table 4.4-6. Ingest History Log Event Trace Diagram (1 of 2)**

Step	Service	Description
1	Constructor	Operator selects the Ingest History Screen from the GUI interface and instantiates an InHistoryLogMonitor object to read input from the operator
2	CheckPrivilege()	Verify the Operator's privilege for log viewing

**Table 4.4-6. Ingest History Log Event Trace Diagram (2 of 2)**

Step	Service	Description
3	ReceiveMsg()	Operator enters criteria information for the search
4	ProcessRequest()	Request to perform log viewing
5	GetHeaderData()	Based on the criteria provided by the Operator, invoke the service to query the Ingest InRequestSummary Header according to the Operator's specification
6	DisplayResults()	Format and display the log results to the Operator's screen
7	GetData()	Based on Operator's selection of a single Request, get detail File Group information from InRequestSummary data
8	DisplayResults	Format and Display Request File Group information



**Figure 4.4-6. In\_Ingest\_History\_Log\_Viewing\_Event\_Trace\_Diagram Dynamic Model**

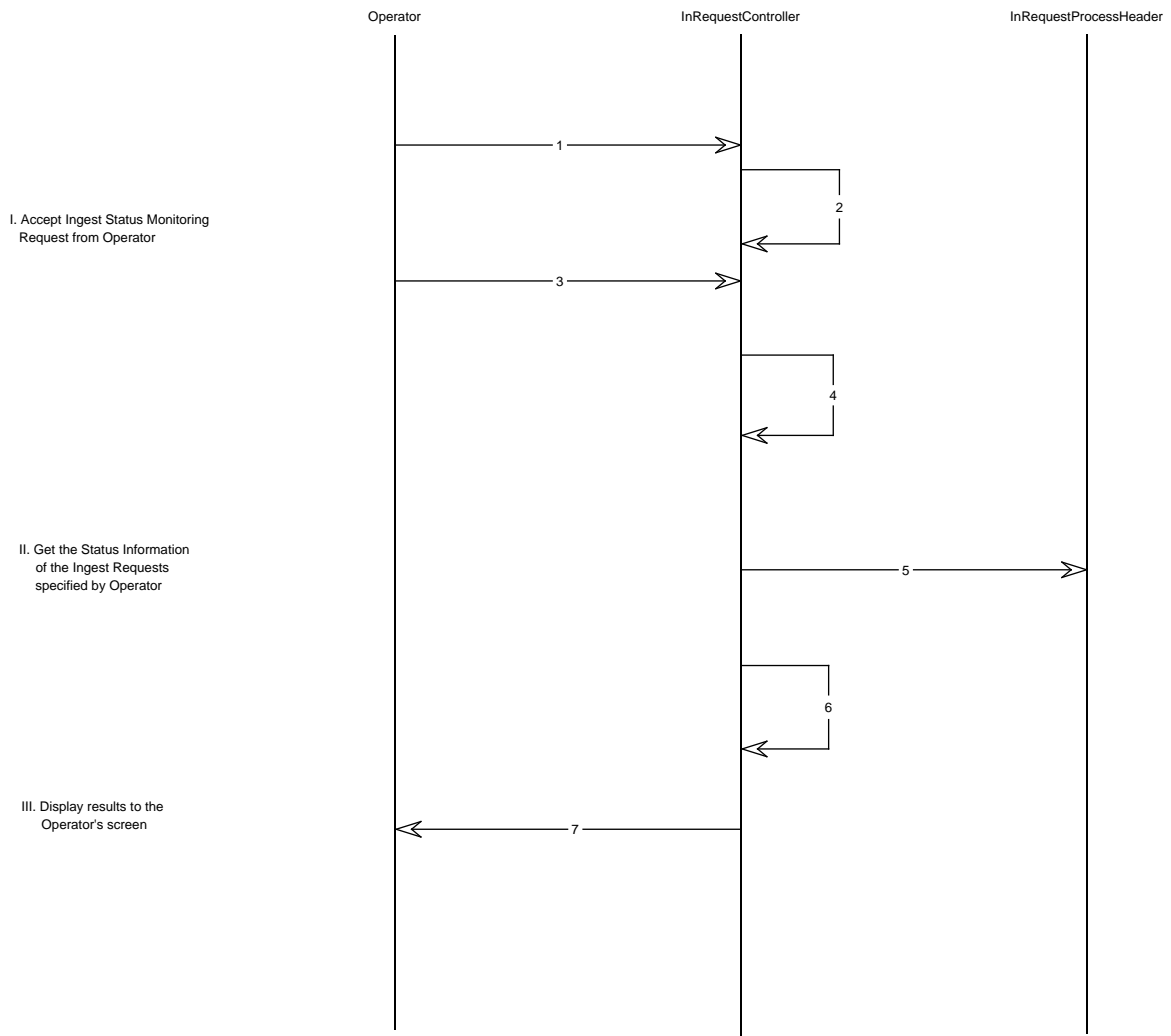
#### 4.4.7 Operator Ingest Status Monitoring Scenario

The ECS system provides operations personnel with the capability to monitor the status of the ingest requests that are in progress. The operator has the capability to look at the status of all requests or at only specific requests. An InStatusMonitoring screen will be configured on the Operator's GUI Interface to accept the ingest status monitoring request and the criteria specification from the operations personnel via the GUI Interface and invoke the appropriate service to get and display the request states information to the operations personnel's GUI screen.

The following list describes the Operator Ingest Status Monitoring scenario using object classes. Figure 4.4-7 is the corresponding event trace diagram. The numbers in the following list refer to the steps in the diagram.

**Table 4.4-7. Operator Ingest Status Monitoring Event Trace**

Step	Service	Description
1	Constructor	Operator selects the Ingest Request Monitor and Control option from the GUI interface and instantiates a InRequestController object to read input from the operator
2	CheckPrivilege()	Verify the Operator's privilege for request status viewing
3	ReceiveMsg()	Operator enters criteria information for the request search and is read in as a status monitor request
4	ProcessRequest()	Request to perform the request status monitoring
5	SearchRequest()	Based on the criteria provided by the Operator, search for the ingest request
6	Format Results()	Format Results returned from Search
		Repeat step 6 for every request that satisfies the criteria specification
7	DisplayResults()	Display the states of ingest requests to the Operator's screen



**Figure 4.4-7. In\_Ingest\_Operator\_Status\_Monitoring Event Trace Diagram**

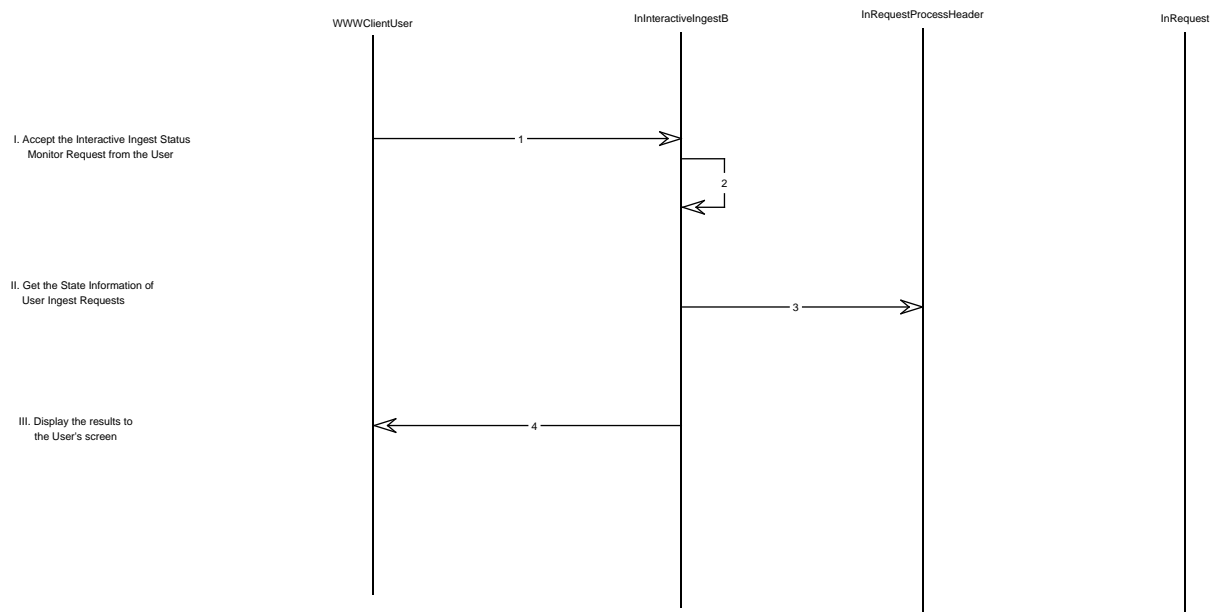
#### 4.4.8 Interactive Ingest Operator Status Monitoring Scenario

The ECS system provides the science users with the capability to monitor the status of the user's on-going ingest requests. A StatusMonitoring HTML Form will be available for user selection from the Interactive Ingest HTML Form Interface (see Section 4.4.4).

The following list describes the Interactive Ingest Status Monitoring scenario using object classes. Figure 4.4-8 is the corresponding event trace diagram. The numbers in the following list refer to the steps in the diagram.

**Table 4.4-8. User Ingest Status Monitoring Event Trace**

Step	Service	Description
1	NA	User selects the Ingest Request Status Monitoring option from the Interactive Ingest Interface Request Form accessed on the WWW
2	NA	Request information is filled in by the user (user name and data type are required)
3	GetStatus()	The Interactive Ingest Forms Script will access request data base to determine current status of all requests from the user for the specified data type (if data type is not specified, all requests for the user are retrieved)
4	DisplayResults()	Request(s) status are displayed to user



**Figure 4.4-8. In\_Interactive\_Ingest\_Status\_Monitoring\_Event\_Trace Diagram**

#### 4.4.9 Operator Request Control Scenario

The ECS system provides operations personnel with the capability to update an on-going ingest request. The updates include: change priority, cancel, suspend, and resume. An InRequestController screen will be configured on the Operator's GUI Interface to accept the update request from the operations personnel via the GUI Interface and invoke the appropriate service to perform the request update and display the update results to the operations personnel's GUI screen.

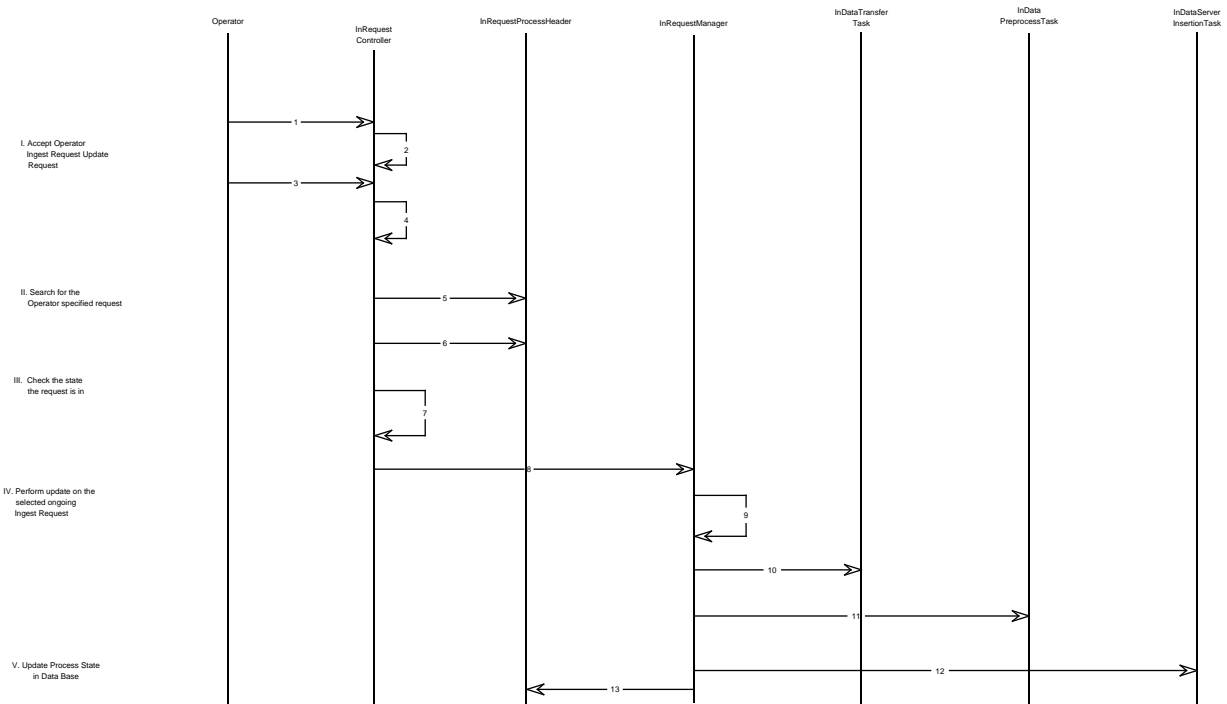
The following list describes the Operator Request Update scenario using object classes. Figure 4.4-9 is the corresponding event trace diagram. The numbers in the following list refer to the steps in the diagram.

**Table 4.4-9. Operator Request Update Event Trace (1 of 2)**

Step	Service	Description
1	Constructor	Operator selects the Request Control option from the GUI interface and instantiates an InRequestController object to read input from the operator
2	CheckPrivilege()	Verify the Operator's privilege
3	ReceiveMsg()	Operator enters request update information and is read in as a request update request
4	ProcessRequest()	Request to perform the request update
5	SearchRequest()	Based on the criteria provided by Operator, search for the ingest request
6	UpdateState	If Cancel, Suspend or Resume has been selected, update process state to cancelling, suspending or resuming
7	ProcessState	Based on State, determine where control/request should be sent. If a hard media ingest is currently being transferred from media, notify the operator to use Hard Media GUI control box to control the request
8	ChangePriority(), Cancel(), Suspend(), or Resume()	Send control request to InRequest Manager for processing
9	ProcessChange	For cancel, suspend, or resume, set flag to notify appropriate pthread that a cancel, suspend, resume or priority change should be performed for change priority request. Use OODCE pthread control capability to increase the priority of the Ingest pthread
10	CancelTransfer(), SuspendTransfer(), or ResumeTransfer()	If the request is in Data Transferring state, ask the DataTransferTask object to perform the specified service: CancelTransfer(), SuspendTransfer(), or ResumeTransfer()
11	CancelPreprocess(), SuspendPreprocess(), , or ResumePreprocess	If the request is in the Data Preprocessing state, ask the DataPreprocessTask object to perform the specified service: CancelPreprocess(), SuspendPreprocess(), or ResumePreprocess()

**Table 4.4-9. Operator Request Update Event Trace (2 of 2)**

Step	Service	Description
12	CancelInsert(), SuspendInsert(), or ResumeInsert()	If the request has already been sent to the Data Server for insertion, ask the DataServerInsertionTask object to perform the specified service: CancelInsert(), SuspendInsert(), or ResumeInsert()
13	UpdateState()	Update process state in the InRequestProcessHeader database file. The updated state will be displayed on operator's request monitor console



**Figure 4.4-9. In\_Ingest\_Operator\_Request\_Update\_Event\_Trace Diagram**

#### 4.4.10 Preprocessing Scenario

The Ingest Subsystem provides services to preprocess all types of ingested data. Preprocessing includes extraction of metadata, conversion of metadata into a standard ECS format, metadata range/field checking, and converting/reformatting science and ancillary data.

The following scenario describes the interaction between the Preprocessing CSC and external classes from the initiation of data granule preprocessing to insertion of the preprocessed data into the Data Server Subsystem. The scenario applies to a data type granule which consists of a separate science and metadata file or a single data file where the metadata is embedded within the science data. Other categories of data type granules follow a similar scenario. Figure 4.4-10 is the corresponding event trace diagram.

**Table 4.4-10. Preprocessing Event Trace (1 of 2)**

Step	Service	Description
1	Preprocess	Initiate preprocessing task
2	InDataPreprocessList	Create an initial list of files to be inserted into the Data Server Subsystem
3	Preprocess	Initiate preprocessing on data granule
4	GetDTInfo	Obtain a list of file types associated with specific data type
5	GetNext	Get next file off of Input List created by Request Processing CSC
6	GetFileType	Obtain file type of file
7	GetFTInfo	Get information characterizing file type
8	InScienceData	Assume file type is science data, and instantiate correct InScienceData specialization
9	Preprocess	Execute required preprocessing on science data
10	AddtoList	Add new file to Data Server insertion list
11	GetNext	Get next file off of Input List created by Request Processing CSC
12	GetFileType	Obtain file type of file
13	GetFTInfo	Get information characterizing file type
14	DsCIDescriptor(GIClient &, UR &, DsSdTypeID &	Create appropriate DsCIDescriptor object
15	GetMCF(ostream &)	Access target metadata configuration file
		Create a file object to store target metadata configuration file
16	InMetadata	Assume file type is metadata, and instantiate correct metadata specialization
17	Preprocess	Execute required preprocessing on metadata
18	PGS_MET_INIT	Initial metadata tool and load target metadata configuration file
19	PGS_MET_GetNext	Get next target metadata parameter
20	GetParInfo	Get information correlating target metadata parameter with source parameter name and location of required data
21	GetParVal	Perform necessary functions to obtain required value out of source metadata file
22	Read	Extract required value from source metadata file

**Table 4.4-10. Preprocessing Event Trace (2 of 2)**

Step	Service	Description
23	PGS_MET_SET	Set value in target metadata configuration file
24	Repeat Steps 20-25	
25	PGS_MET_WriteFile	Write final target metadata configuration file into a PVL file
26	Validate	Validate metadata PVL file
27	AddtoList	Add new file to Data Server insertion list
28	SendInsert	Initiate data insertion
29	Refer to DID-313, Internal Interfaces, Section 4.2.12, Insert_Data_SP12, for data insertion details	Open a data server session



## 4.5 CSCI Structure

Table 4.5-1 shows the Computer System Components (CSCs) that comprise the Ingest CSCI. Details for each CSC are provided in the following paragraphs. Table 4.5-2 shows the classes making up each CSC.

Figure 4.5-1 shows the interactions of Ingest CSCs. Non-Ingest components are indicated by shading.

**Table 4.5-1. Ingest CSCI Components**

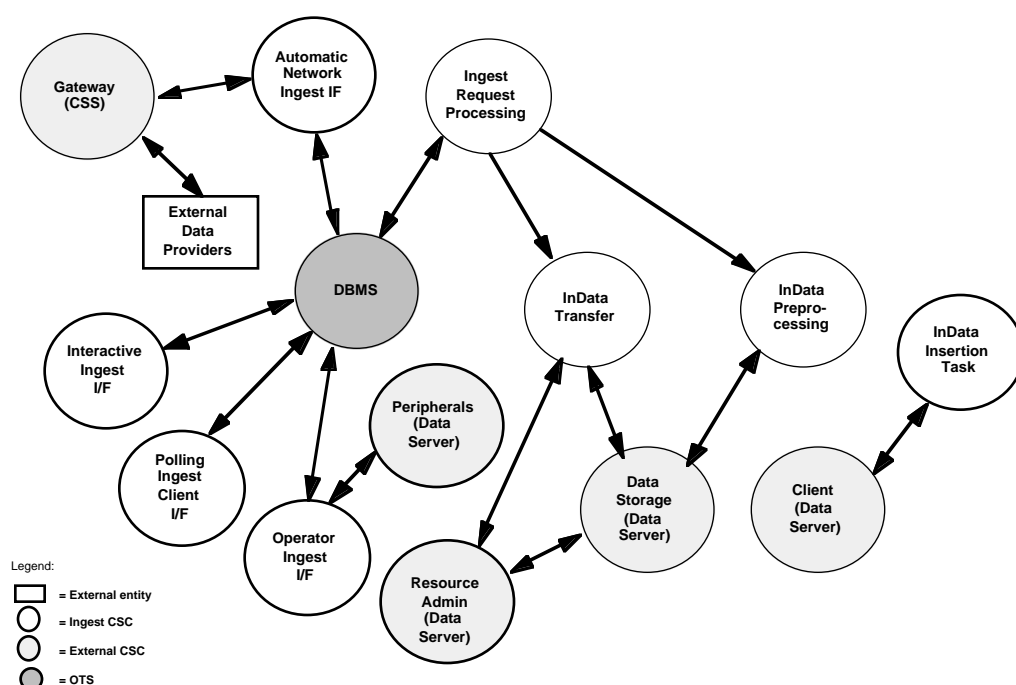
CSC	Description	Type (Custom=DEV; off-the-shelf=OTS)
Automatic Network Interface	Communicates with the CSS Gateway	DEV
Polling Ingest Client Interface	Polls for data files or Delivery Record files in an agreed location	DEV
Ingest Request Processing	Moderates ingest processing steps	DEV
Ingest Data Preprocessing	Performs required preprocessing and interface with the Data Server for data insertion	DEV
Ingest Data Transfer	Transfers data from source to ECS staging space	DEV
Operator Ingest Interface	GUI screens allowing operations staff ingest of hard media, ongoing ingest request status monitoring, completed ingest request information viewing, ingest request controlling (e.g., canceling request; suspending and resuming a request or set of requests; and changing request priority), and ingest threshold controlling (i.e. to view or to set the threshold).	DEV
Interactive Ingest Interface	HTML Forms allowing users to ingest approved data and to perform ongoing ingest request status monitoring	DEV
Ingest DBMS	Data Base Management System used to store and provide access to the Ingest History Log and other ingest internal data	OTS
Ingest Administration Data	Provide services to access the History Log and administrative information of the Ingest Subsystem	DEV
Peripheral Software	Provide all media peripheral access software and operator administration functions for ingest peripherals	Reuse
Viewing Tools	Tools to allow displaying of ingested data for validation (analysis) purposes	Reuse
Data Storage Software	Software to store Level 0 data on working storage and repository storage (for one year)	Reuse
Resource Administration	Operator administration software to manage and control the Data Storage Software	Reuse
Client	Provides Science Data Server client interface services.	Reuse

**Table 4.5-2. Ingest CSC Component to Class Mappings (1 of 2)**

CSC	Classes
Automated Network Ingest Interface	InServerExtRPC_C InServerExtRPC_S InServerIntRPC_C InServerIntRPC_S InSession InSessionExtRPC_C InSessionExtRPC_S InSessionInfo InRequestInfo InMessage InShortDAA InLongDAA InShorDDN InLongDDN InDAN
Polling Ingest Client Interface	InPollingIngest Session
Ingest Request Processing	InRequest InRequestManager InRequestManager_C InRequestManager_S InGranuleServer_C InGranuleServer_S InGranulesAsync_C InGranuleAsync_S InGranuleMessage InRequestProcessData InRequestProcessHeader InRequestFileInfo
Ingest Data Preprocessing	InData PreprocesTask InMetadata InBOMetadata InBOBinMetadata InPVMetadata InSDMetadata InHDFMetadata InScienceData InFDFData InSnowIceData InTOMSDData InGRIBData InReformatData InDataTypeTemplate InFileTypeTemplate InSourceMCFTemplate InDataPreprocessList InDataServerInsertionTask
Ingest Data Transfer	InDataTransferTask InResourceIF InFile

**Table 4.5-2. Ingest CSC Component to Class Mappings (2 of 2)**

CSC	Classes
Operator Ingest Interface	InGUISession InIngestMainWindow InMediaIngest InMediaCheckIn InRequestController
Interactive Ingest Interface	InInteractiveIngest
Ingest Administration Data	InRequestSummaryData InRequestSummaryHeader InExternalDataProviderThreshold InPollingThreshold InDBAccess



**Figure 4.5-1. Ingest CSC Interaction**

The Automated Network Ingest Interface CSC sets up ingest sessions with external data providers (e.g. TSDIS, SDPF and Landsat-7 LPS), via the CSS TCP/IP to OODCE Gateway. External data providers submit Data Availability Notices (DANs) to request data ingest. Polling Ingest Client Interface components poll accessible file system locations to detect data to be ingested; the component submits an equivalent DAN. The Interactive Ingest Interface CSC allows an authorized science user, document supplier or operator to create and submit a DAN interactively. The Automated Network Interface CSC, the Polling Ingest Client Interface CSC and the Interactive Ingest Interface CSC all submit ingest requests (containing DAN data items) to the Ingest Request Processing CSC. The Ingest Request Processing CSC manages subsequent request

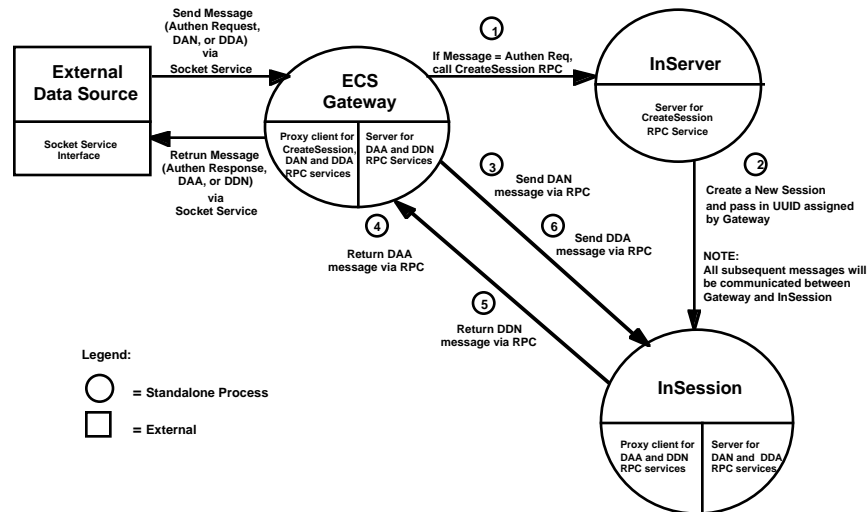
processing. The Ingest Request Processing CSC invokes the Data Transfer CSC to transfer data from external locations. The Ingest Request Processing CSC invokes the Data Preprocessing CSC to preprocess ingested data (e.g., validate metadata parameters) and then invokes the InDataTransfer Task to insert data into the Data Server.

The shaded Data Server CSCs--Peripherals, Data Storage, Client, and Resource Administration--provide data storage and peripheral access services. In the case of Level 0 data ingest, the Data Server CSCs are reused in the Ingest Subsystem and implemented on Ingest Subsystem hardware. For non-Level 0 data, the Ingest and Data Server CSCs are implemented on Data Server hardware.

#### 4.5.1 Automatic Network Ingest Interface

The Automatic Network Ingest Interface CSC provides the fundamental capabilities to ingest data into the ECS system. The CSC can be tailored to fit in a specific interface and is comprised of the InServer, InServerExtRPC, InSession and InSessionExtRPC object classes. This CSC corresponds to the "Generic Ingest Client Shell" CSC described in the SDPS System Design Specification. Figure 4.5-2 illustrates the interaction between an external data source and the Automatic Network Ingest Interface CSC.

The InServerExtRPC and InSessionExtRPC object classes define the OODCE Remote Procedure Calls (RPCs) interface. The client is responsible for invoking the RPCs to request Ingest services. The client initializes a connection to ECS Ingest Subsystem by invoking the CreateSession() RPC of the InServerExtRPC object class. This RPC creates an Ingest Session and establishes a connection with the external source. All the subsequent interactions with the external source are defined in the InSessionExtRPC object class.



**Figure 4.5-2. Automated Network Ingest Interface**

If the external data source is not a OODCE client, there will be an ECS Gateway that will receive the TCP/IP socket service calls from the external data source and translate the socket service calls into OODCE RPC calls. The ECS Gateway will be supplied by the CSS Subsystem with support

from Ingest. When the ECS Gateway receives an Authentication Request from the external source, it authenticates the client and invokes the CreateSession RPC of the InServerExtRPC. A session is created for the client and all other subsequent messages received over the TCP/IP sockets from that client are mapped to an RPC and forwarded to the associated InSessionExtRPC object class.

The InServer object class is instantiated from the main program Ingest Server. It sets up as a server and listens for incoming RPCs. It provides a single point of entry to the Ingest Subsystem for all Automated Network ingest interfaces. When the external data source invokes the CreateSession() RPC, an Ingest Session is established and linked with the external data source by the Ingest Server. The InServer object class is responsible for managing all ingest sessions processing under the server (i.e., keeping track of sessions by adding the session to its list when a new session is created and deleting the session from its list when the session is terminated). A single session is created for a given client. Upon termination of the InSession, the InServer deletes the associated session and client information from its list. The InServer object class will be configured as a standalone program initiated at system startup on the Ingest Client HWCI.

The InSession process is invoked by the CreateSession RPC of the InServerExtRPC object class. The InSession object class is instantiated from within the process, after which the session is setup as a server and listens for incoming RPCs from its client. The responsibility of the InSession object class consist of 1) managing multiple requests from a single client, where each request corresponds to a DAN message (RPC) with unique DAN sequence number, 2) instantiating an InDan object which it uses to checkpoint DAN information in the InRequestProcess Header, InRequestProcess Data and InRequestFileInfo Ingest Database classes, 3) sending a CreateRequest RPC call to InRequest Manager to begin request processing, 4) sending outgoing Data Availability Acknowledgments (DAA) to acknowledge the receipt of the DAN and Data Delivery Notifications (DDN) to inform the client of the final status of their Ingest request, 5) cleaning up the request information upon receipt of a Data Delivery Acknowledgement (DDA) from the client, and 6) terminating the connection with the client upon completion of all requests. In addition, the object class provides services to suspend and resume a session. The InSession object class will be configured as a standalone program initiated by the InServer object class when the InServer receives the CreateSession RPC.

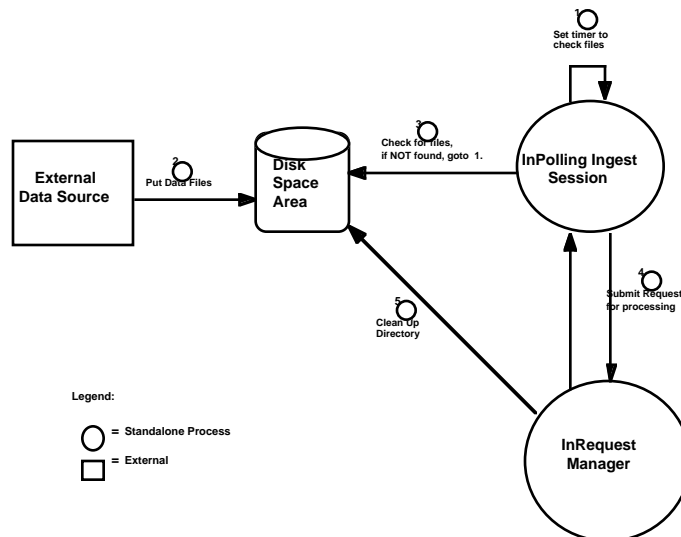
The Automated Network Ingest Interface CSC provides the capability to ingest data in an automated fashion by means of network data transfer into the ECS system. The ingest process is initiated based on a stimulus provided by a DAN from the external interface. The TSDIS, Landsat-7 LPS, SDPF, and SCF interfaces are candidates for the use of the Automated Network Ingest Interface. This CSC corresponds to the "Ingest Clients for each External Interface" CSC described in the SDPS System Design Specification. Figure 4.5-2 shows the public services provided by the object classes of the Ingest Session Manager CSC.

#### **4.5.2 Polling Ingest Client Interface CSC**

The Polling Ingest Session CSC provides ECS with the capability to ingest data from data centers with little or no handshaking. The CSC is comprised of the InSession and InPollingIngestSession object classes. This CSC corresponds to the "Ingest Clients for each External Interface" CSC described in the SDPS System Design Specification. Figure 4.5-3 provides a pictorial overview of the Ingest Polling processing.

The responsibilities of the InPollingIngest Object Class consist of 1) creating the proper polling request, 2) detecting new files of interest at a tunable period of time in an external or local disk location, 3) creating the InRequest Object Class (a component of the Ingest Processing CSC (refer to Ingest Request Processing CSC for ingest details), 4) creating a unique identifier for the request and adding requests to the InRequestProcessHeader, InRequestProcessData and InRequestFileInfo Object Classes, and 5) reporting the status of its ongoing ingest requests.

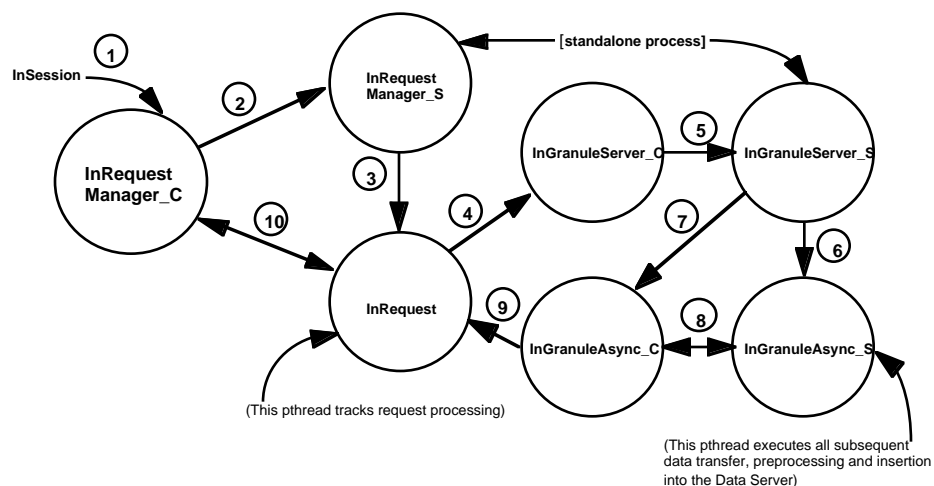
The NESDIS, GSFC Data Assimilation Office (DAO), LaRC DAAC, GSFC DAAC, ACRIM, SAGE-III, and EDOS interfaces are candidates for using the polling ingest protocol. The InPollingIngestSession (Files) Scenario addresses the NESDIS and DAO interface, while the InPollingIngestSession (Delivery Record) Scenario addresses the other interfaces. The InPollingIngestSession (Delivery Record) Scenario proposes that the Ingest Subsystem detect and read information contained in a Delivery Record. The Delivery Record has identical structure to that of the Data Availability Notice (DAN) used in the Automated Network Ingest Interface. The InPollingIngestSession (Files) Scenario proposes that the Ingest Subsystem detect and read information contained in a remote directory. The functionality to initiate both scenarios is contained in the Polling Ingest Session CSC.



**Figure 4.5-3. Ingest Polling CSC**

### 4.5.3 Ingest Request Processing CSC

The Ingest Request Processing CSC is the core component of the Ingest Subsystem. It manages the ingest request traffic and the processing of the ingest requests. The CSC provides the capability to process multiple ingest requests concurrently. The CSC is responsible for tracking the ingest requests and coordinating the ingest processing, which is comprised of transferring data, performing data preprocessing, and sending an insertion request to the appropriate Data Server. The CSC is composed of InRequest, InRequest\_C, InRequest\_S, InRequestProcessHeader, InRequestProcessData, InRequestFileInfo, InSystemThreshold, InExternalDataProviderInfo, InGranuleServer\_C, InGranuleAsync\_C, InGranuleServer\_S, InGranuleAsync\_S, InRequestManager, InRequestManager\_C, and InRequestManager\_S object classes. This CSC is a new CSC not described in the SDPS System Design Specification.



**Figure 4.5-4. Ingest Request Processing CSC**

The IngestRequestProcessing CSC follows the object factory model approach. In the factory model a "factory object" is established to create other objects based on a client request. The factory object provides a client and server component. For the Ingest Processing CSC the factory object is InRequestManager\_S, which operates as a "server". The client proxy object is InRequestManager\_C. It is linked with the "client".

InRequestManager\_S creates InRequest objects. The InRequest object using the Server Request Framework (SRF) sends processing requests to InGranuleServer\_S for asynchronous granule processing by InGranuleAsync\_C objects. The InGranuleServer\_S and InGranuleAsync\_S objects provide an object factory model approach to granule processing. InGranuleAsync\_S objects handle all subsequent data preparation and insertion into the Data Server. InRequestManager\_S allocates concurrent processing volume thresholds for each request it instantiates. These concurrent processing volume thresholds will be used by the InRequest object to throttle the instantiation of the granule objects which perform the data transfer and preprocessing functions. This throttling at the granule level permits multi-granule requests to begin processing even though total request processing requirements exceed current threshold limits. Once InRequest objects are created, they communicated directly with InSession to return DDN status (Automated Network Ingest Requests only). Figure 4.5-4 shows the interaction of the generic object classes involved in the Ingest Request Processing CSC.

At system startup, the CSC is configured as multiple standalone programs, including a single InRequestManager and one or more InGranule Servers. The single copy of InRequestManager runs on the Ingest Client HWCI (or, at sites with no Ingest Client HWCI, or the Data Server subsystem equivalent) with only InRequestManager\_S instantiated. InGranule Servers run on the Ingest Client HWCI and on any Data Server HWCI where granule processing will be performed. Each InGranuleServer Task is started up with only a single InGranuleServer\_S object instantiated. Locating InGranule Server tasks on Data Server as well as the Ingest HWCI promotes efficient handling of transferred data by allowing Ingest to preprocess data where it has been staged (transferred). The InRequestManager\_C objects are instantiated as a client proxies.

InRequestManager\_C is invoked (Step 1 of Figure 4.5-4) which sends an RPC request to InRequestManager\_S (Step 2) which in turn creates an InRequest object as a separate pthread (Step 3). One pthread is created for each InRequest object. The distributed object reference (OID) for InRequest is returned to InSession. InSession handles subsequent communications with InRequest. In particular, InRequestManager\_S invokes the ProcessRequest service of InRequest which, for each granule in the request, sends a granule process request to InGranuleServer\_C object (Step 4). InGranuleServer\_C builds a process granule request which is sent to InGranuleServer\_S via SRF (Step 5). The InGranuleServer\_S constructs both the InGranuleAsync\_S object (Step 6) and the InGranuleAsync\_C object (Step 7). The InGranuleAsync\_S object handles the data transfer, preprocessing and insertion of a granule into the Data Server. All InGranuleAsync\_S objects for a request are processed in parallel (asynchronously). When a granule is complete (inserted into the Data Server), the InGranuleAsync\_S object notifies the InGranuleAsync\_C object of the completion (Step 8). Once all granules for a request have completed, InRequest object is notified (Step 9). InSession waits for the return (via InRequestManager\_C) of status upon completion of the data insertion (Step 10).

The client (InSession) checkpoints persistent context information about the OODCE connections established above. In the event of a failure of the client, the OODCE context information is used to reestablish the connection and receive return status.

The InRequest object attributes (including OODCE context information) are checkpointed in a DBMS using InRequestProcessHeader and InRequestProcess Data objects. In the event of a failure in InRequestManager\_S or in InRequest, the InRequestManager\_S object (after process restart) restores all ongoing requests using checkpointed information. Each InRequest object subsequently restores its contents, including OODCE context, from the checkpointed information.

The InRequest object contains information needed to start ingest processing for all granules in a request. The InRequest object stores the state mode (i.e., "active" or "suspended") of the ingest request. The InGranuleAsync\_S objects manage each granule's state (i.e., "transferred", "preprocessed", "submitted to Data Server", and "Data Server insertion complete") of each data granule specified in the request. State transitions are reported to the corresponding InGranuleAsync\_C object which decrements the requests' concurrent processing volume statistics. Granules being delayed by InRequest throttling algorithms will be re-evaluated by InRequest and instantiated if threshold limits are no longer exceeded.

#### **4.5.3.1 Evaluation of AutoSys to Support Ingest Request Processing**

The AutoSys COTS product was evaluated for integration with the Ingest Request Processing CSC. The result of the investigation was to reject the use of AutoSys. AutoSys has three main strengths: 1) job execution on a pre-established job schedule, 2) concurrent job processing and 3) job load balancing across processors. Each of these strengths and their applicability to Ingest requirements are discussed in the following paragraphs.

AutoSys's main strength is job execution on a pre-established schedule. This strength cannot be directly utilized by Ingest, since Ingest clients are not required or expected to submit ingest requests on a pre-established schedule.

AutoSys's second strength is concurrent job processing. Ingest achieves concurrent job processing via the object factory model approach discussed in detail in the preceding section. Ingest's factory

model utilizes OODCE pthreads to provide concurrent processing of requests. The object factory approach is further supported by the ECSs Server Request Framework (SRF) which Ingest is utilizing to provide concurrent processing of each granule within a request. AutoSys does not use pthreads; instead, processing is performed in standalone processes. This requires more system resources than using pthreads.

AutoSys's third strength, job load balancing across processors, cannot be utilized by Ingest since the data type of an ingest request will dictate on which Data Server instance processing will occur. Each Data Server instance is pre-assigned a group of data types. In particular, all Level 0 data archiving will utilize the Data Server instance running on the ICLHW.

Despite the mismatch of strengths to requirements, AutoSys was investigated to see if it could offer Ingest a COTS solution to job queue management. Ingest requests may require queuing, if resources are not available when the Ingest request is received. The AutoSys investigation determined that AutoSys requires a significant amount of custom software support to manage a job queue. The ECS Planning subsystem, current users of AutoSys, have elected to, via custom software, reschedule and resubmit to AutoSys any job which fails to execute due to resource limitations. AutoSys therefore, does not offer a solution to job queue management which could be readily used by Ingest. Ingest therefore will rely on Data Server resource allocation queues as well as the Ingest request throttling capability, discussed in the preceding section, to provide job queues when necessary.

#### **4.5.4 Ingest Data Transfer CSC**

The Ingest Data Transfer CSC provides services to the Ingest Client CSCI to facilitate transfer of data files into the Ingest Subsystem (on a file by file basis), to collect information on individual ingested files, build lists of file objects to be ingested, and group files with valid data types. In addition, the CSC allows the operator 1) to cancel the data transfer, 2) to put the data transfer on hold, and 3) to resume the data transfer that was previously put to hold. This collective set of services within the Ingest Data Transfer CSC is provided by the InDataTransferTask, InTransferredData, and InFile object classes. The CSC is configured as a standalone program initiated by the Ingest Request Processing CSC for every set of data granules to be transferred as supplied by the Ingest Request Processing CSC. This CSC is a new CSC not described in the SDPS System Design Specification.

The InDataTransferTask Object Class manages data transfer associated with a specific ingest request. The InDataTransferTask Class provides services to instantiate files, invoke data transfers (file by file), and allocate storage space via the InResourceIF Object Class.

The InFile Object Class provides services to obtain file information on files associated with a specific transfer and to group a list of files with valid data types. This object class will provide services to the InDataTransferTask Object Class to obtain data type information.

The InResourceIF Object Class serves as the interface to the resource/device services by which the Data Server Subsystem provides. The Object class interfaces with the Storage Resource Management and the Data Distribution CSCIs of the Data Server Subsystem to perform these services. Refer to the Data Server volume for details.

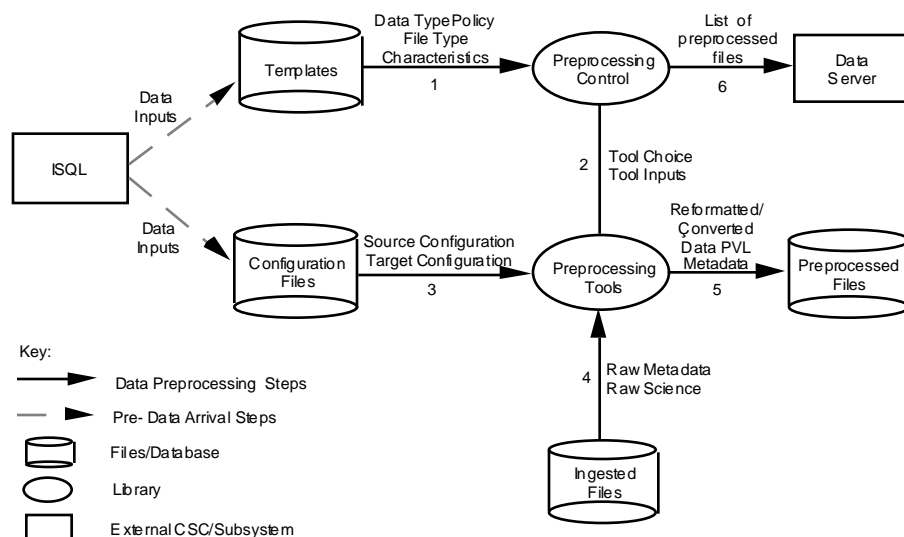
#### 4.5.5 Ingest Data Preprocessing CSC

The Ingest Data Preprocessing CSC provides services to perform required preprocessing of data and subsequent insertion of the data into the appropriate Data Server. The preprocessing of data consists of converting the data (if needed), extracting the metadata into the standard ECS metadata format (if needed), performing required metadata existence and parameter range checks, and updating the metadata with ingest specific metadata (e.g., start and stop date/time for ingest). In addition, the CSC is responsible for updating the request state, tracked by the Ingest Request Processing CSC, whenever its state changes; and accepting request cancellation, suspension, and resumption requests from the Ingest Request Processing CSC. This CSC corresponds to the "Translation Tools" and "Data Compression/ Decompression Tools" CSCs described in the SDPS System Design Specification.

The CSC depends on templates and configuration files to maintain data/file type policy and source/target format information. These templates and configuration files supply necessary information to other classes within the CSC in order to perform the required preprocessing for a specific data type granule. The configuration files specify the input format and target format for specific data/file types. Preprocessing classes rely on these files to specify how an ingested file is organized upon arrival (source metadata configuration file), and how the data should be organized before insertion (target metadata configuration file) into the data server. Most importantly, configuration files can be modified with no software recompilation or modification within the ingest preprocessing CSC. Source metadata configuration file modifications can be executed through the Sybase ISQL. Target metadata configuration files are supplied by the Data Server Subsystem.

Each data type may consist of different file types (e.g., metadata, science, calibration). The constituents of each data type are defined in a data type template. This template has a record for each ingest data type specifying the file types that make up the data type. In addition, each file type has unique characteristics. For instance, each different file type uses different metadata configuration files. This type of information is contained in a file type template. These templates may be modified through the Sybase ISQL. New entries (types) can be added to each of the templates without recompilation or modification within the preprocessing CSC.

The utilization of configuration files and templates and their interaction with preprocessing processes is illustrated in Figure 4.5-5, "Ingest Preprocessing CSC Data Flow". The preprocessing control library obtains information from the templates to determine which preprocessing tools are required (for a specific data/file type) and what the necessary inputs are for the chosen tools (1). With the acquired information, preprocessing control can invoke the appropriate preprocessing tools (2). The configuration files aid the preprocessing tools in determining how the ingested input data is organized and the target format of the data (3). The preprocessing tools provide the functionality to obtain the values of required parameters directly from the ingested files (4). The preprocessing tools transform the ingested files into final products (5). Upon completion of preprocessing, a list (or lists) of files is presented to the Data Server Subsystem for subsequent data archival (6).



**Figure 4.5-5. Ingest Preprocessing CSC Data Flow**

The Ingest Preprocessing Object Model (presented in section 4.3) provides an object oriented view of the CSC. This object model presents preprocessing unique classes, as well as classes from other CSCs or subsystems. Therefore, an understanding of the Preprocessing CSC and its interaction with external classes can be viewed in one continuous model. The Ingest Data Preprocessing CSC is composed of the InDataPreprocessTask, InMetadata (and its subclasses), InScienceData (and its subclasses), InDataTypeTemplate, InSourceMCF, InDataPreprocessList, InMetadataTool, and InDataServerInsertionTask object classes. Each class is described in detail in the Preprocessing CSC data dictionary and their respective roles are illustrated in the detailed preprocessing scenario (Table 4.4-10). However, a general understanding of the object model can be gained by flowing through the following high level object-oriented scenario.

A "preprocessing task" is initiated when the InDataPreprocessTask Class is instantiated by the Ingest Request Processing CSC. As a result, the InDataPreprocessTask class controls the preprocessing of a set of files associated with a given data type (on a per granule basis). This set of files is contained in an InDataPreprocessList object. The InDataPreProcessTask utilizes the file list and data type argument to access data/file type specific information from the InDataTypeTemplate and InFileTypeTemplate classes. The acquired information is used to properly instantiate the appropriate data type subclasses (i.e., InPVMetadata). The instantiated data type subclasses then interact with other classes (i.e., InSourceMCF, DsCIDescriptor, InMetadataTool) to produce final products before insertion into the Data Server Subsystem. The InSourceMCF class provides information on a data type basis on how to retrieve applicable parameter values (e.g., the location of parameter, computer data type). The DsCIDescriptor is a data server class which provides services to acquire a target metadata configuration file and validate metadata. Utility tools (provided by PDPS Toolkit) provides services to read target metadata configuration files and write PVL metadata files. After completion of the preprocessing task, the InDataServer InsertionTask Class interacts with the Data Server Subsystem (DsCIESDTRreferenceCollector, DsCIRequest, DsCICommand) to insert the preprocessed data files. Detailed scenarios illustrating interactions

between all classes are provided later in this section. With the basic concept of operations in place, the design principles used to develop the object models and individual classes are presented in the following paragraphs.

The Ingest Subsystem will receive data from a variety of external sources and instruments as illustrated previously in Figure 3.1-1. This variety of instruments and sources produce data types which have different file formats (e.g., HDF, SFDU, GRIB), data formats, metadata parameters, and data organizations (e.g., separate file for metadata vs. embedded metadata). This diverse set of ingest data types necessitates different implementations of preprocessing services based on each individual data type or set of data types. For instance, the implementation of conversion services requires different algorithms for data types which possess different initial and/or target formats. In addition, each ingest data type may make use different preprocessing services. For instance, some data types require metadata extraction (e.g., CERES L0 Data, NMC GRIB data), while other data types may provide a separate file which contains only metadata (e.g., TOMS Gridded Ozone, AVHRR Monthly GVI). Therefore, the first major design goal for the Ingest Preprocessing CSC is to provide the capability to preprocess this diverse set of data types efficiently and adequately. The tailoring of preprocessing services to address this diverse set of data types is done by specializing the data type base classes, InMetadata and InScienceData.

The level of specialization for these data type base classes is limited by the second major design goal, eliminating redundancy. The data type subclasses are designed to address groups of data types (where possible) versus a subclass for each data type. If the data types within each group are similar, the services within these subclasses can be tailored to properly preprocess the individual data types through the arguments of the called services. The values of these arguments, for a specific data type, are obtained through the InFileTypeTemplate. The "Template" Class contains information on how to preprocess each specific data type/file type. Part of the information contained in these classes also defines the argument values (dependent on data type) for subsequent use of data type subclass services.

For example, the InPVMetadataClass is a specialization of the InMetadata Class which contains the appropriate preprocessing service for parameter-value metadata. For different data types within this group (parameter-value metadata), different delimiters may be used to separate a parameter from a value. The delimiter for each specific data/file type is defined in the InFileTypeTemplateClass. This information is passed to the InPVMetadataClass upon instantiation. The InPVMetadataClass services will utilize the input arguments to define the appropriate delimiter for a specific service. Whether the delimiter is a semicolon or equal sign does not effect the design of the service. However, it does effect the execution of the service and the resultant product. The delimiter may differ with each data type and therefore must be defined in the input arguments of the service.

This design also helps fulfill the third major design goal, flexibility and extensibility. The CSC is flexible, since modification of existing data types will be feasible. For example, changes in input metadata structure or required metadata output structure are driven by the metadata configuration files. The Data Server subsystem will provide services to access "target" metadata configuration files. These files will dictate the metadata requirements for each data type. Through an iterative process, the preprocessing CSC will retrieve the input values from the input metadata file(s). The InSourceMCF class, provides correlation between the target parameter names in the target

metadata configuration file with access information on the input metadata file. The InMetadata and InScienceData classes utilize these metadata configuration files to produce final products.

The system is extensible, since new data types may be added without major design change. Not only does the basic design often not have to change to accommodate new data types, but some of the preprocessing subclasses may be used for future data types. The implementation of broad based specializations, provides a greater probability that a new data type will fit into an existing data group versus creating a new class for that data type.

The following paragraphs will discuss general preprocessing requirements and how each requirement applies to some of the ingest data types listed in the tables. Table 4.5-3 summarizes ECS data type/set preprocessing requirements for all Release A data types. Items marked as TBR are to be resolved post-Release B CDR with the data providers and processing teams. Table 4.5-4 summarizes ECS data type/set preprocessing requirements for known Release B data types. (Note: 1. Additional ancillary data sets required for Release B processing are currently under review by the Data System Working Group (DSWG) and the ad hoc working group for processing. Table 4.5-4 will be updated post-CDR. 2. All Release A ancillary data continue to be required at Release B).

**Table 4.5-3. Release-A Ingest Data Type Preprocessing Requirements (1 of 2)**

<b>Data Set(s)</b>	<b>File Format</b>	<b>Convert</b>	<b>Reformat</b>	<b>Field/Range Checking</b>	<b>Embedded Metadata</b>	<b>ICD</b>
CERES Level 0 Data	SFDU	No	No	Yes	No	1
LIS Level 0 Data	SFDU	No	No	Yes	No	1
TRMM H/K Data	SFDU	No	No	Yes	No	1
TRMM Orbit Data	SFDU	No	No	Yes	No	1
LIS SCF Constants and Coefficients	Binary/ASCII	No	No	Yes	No	2
CERES SCF Constants and Coefficients	Native/HDF	No	No	Yes	No	2
CERES SCF Generated Ancillary Data Files	Native/HDF	No	No	Yes	No	2
TSDIS Level 1a Data	SFDU	No	No	Yes	No	3
TSDIS Level 1b-3b Data	HDF	No	No	Yes	Yes	3
NMC-MRF	GRIB	Yes - to HDF	Yes	Yes	Yes	5
NMC-ETA	GRIB	Yes - to HDF	Yes	Yes	Yes	5

**Table 4.5-3. Release-A Ingest Data Type Preprocessing Requirements (2 of 2)**

Data Set(s)	File Format	Convert	Reformat	Field/Range Checking	Embedded Metadata	ICD
NMC-FNL	GRIB	Yes - to HDF	Yes	Yes	Yes	5
GPCP Data	Native	No	No	Yes	Yes	7
GPCC Data	Native	No	No	Yes	No	7
AVHRR Global Analyzed Field	Binary Direct Access I/O	No	No	Yes	No	4
Snow/Ice Cover (EDR)	Intermediate Database	Yes - to HDF	Yes	Yes	No	4
AVHRR Weekly General Vegetation Index	Binary Raster Data	No	No	Yes	No	4
TOMS Gridded Ozone	HDF	No	No	Yes	No	8
SAGE-II Stratospheric Optical Depth	HDF	No	No	No	No metadata exists	6
SAGE-II Stratospheric Ozone	HDF	No	No	No	No metadata exists	6
ISSCP DX,D1,D2 radiances	TBR	No	No	No	No	6
SSM/I Level 1b	Native	No	No	Yes	Yes	7
Layer/Level Ozone	Native	No	Yes	Yes	No	4

Note: Digital elevation map and surface map of vegetation are static data sets and do not require ingest preprocessing.

1. ICD between ECS and the SDPF (209-CD-025-002)
2. ICD between ECS and Science Computing Facilities (209-CD-005-004)
3. ICD between ECS and TSDIS (209-CD-007-003)
4. ICD between ECS and the NOAA ADC (209-CD-006-004)
5. ICD between ECS and the GSFC DAAC for the ECS Project (209-CD-008-003)
6. ICD between ECS and LaRC DAAC (209-CD-010-002)
7. ICD between ECS and MSFC DAAC (209-CD-009-002)
8. ICD between ECS and Version 0 System

**Table 4.5-4. Release-B Ingest Data Type Preprocessing Requirements**

Data Set(s)	File Format	Convert	Reformat	Field/Range Checking	Metadata Extraction
NCEP Global 4D Assimilation	GRIB	TBR	TBR	Yes	Yes
NCEP Reynolds Blended SST	GRIB	TBR	TBR	Yes	Yes
Brightness Temp Product from AMSR	TBR	TBR	TBR	TBR	TBR
NOAA POES HIRS/2 Total Ozone Concentration	TBR	TBR	TBR	TBR	Yes

**Metadata Extraction:** The necessity for metadata extraction depends on each individual data type. Landsat 7 Level 0R data does not require metadata extraction since a separate Landsat 7 metadata file is provided. NMC GRIB data requires metadata extraction since the metadata and science data are provided within one physical data file. SDPF Level 0 data provides some metadata in a separate file (Detached SFDU Header) while other metadata (Data Set File Header(s)) are embedded within the science data set file. TSDIS HDF metadata is contained in an separate HDF object from the TSDIS science data. HDF tools will extract the metadata objects from the TSDIS HDF files. TSDIS SFDU metadata is organized in a similar format to the SDPF metadata. The necessity for extraction and information, related to where the metadata resides within a physical file, can be found in the appropriate InSourceMCF and InFileTypeTemplate objects.

**Conversion:** The implementation of conversion services will depend on the initial format of the data type and the targeted archive format. Currently, the three main ingest formats which have been identified for ECS Release A ingest are the: Standard Format Data Unit (SFDU), Hierarchical Data Format (HDF), and Gridded Binary (GRIB). Since HDF is a standard archive format, data conversion from HDF to other formats during the ingest process is not foreseen. SDPF SFDUs (CERES and LIS) will not be converted since the instrument teams will prepare algorithms to process SFDUs as is. TSDIS Level 1A will remain in SFDU format since the TSDIS and ESDIS Projects have jointly determined that it is not efficient or necessary to convert this data to HDF. The lack of efficiency stems from processing complexities with converting TSDIS Level 1A SFDU data into HDF. The lack of necessity is due to TSDIS Level 1A data not being widely distributed to the User community, and therefore a common format such as HDF for this data is not imperative.

An analysis of external ancillary data sets required by the instrument teams has been performed. The result of this analysis was a project directive mandating that only datasets and/or formats required by multiple teams are to be considered for the development of ancillary data reformatting and conversion capabilities. NMC GRIB data and, SSM/I snow and ice products, meet this criteria and therefore will be reformatted and converted as needed. NMC GRIB data will require data conversion to HDF-EOS. The detailed design for the NMC GRIB conversion is in progress and will be reported upon post-Release B CDR. TOMS Gridded Data already exists in HDF. SSM/I

snow and ice products are currently being analyzed. Other CERES ancillary data, GPI, GPCP data will not be converted.

**Reformatting:** The Ingest Preprocessing CSC will perform reformatting to alleviate inconsistencies in data representations from different external computer systems (e.g., data type definitions, byte ordering, and character representation). HDF data does not have to be reformatted since HDF is a portable file format. An HDF file created on one computer system can be easily read on another computer system without modification. SFDU data will not be reformatted since the data is defined in a byte by byte format. GRIB, and SSM/I Snow and Ice Cover will be reformatted as part of the conversion process.

**Metadata Checking:** Metadata checking will be performed to verify that certain metadata fields exist in the correct form and that the value of certain fields are within limits. The extent of metadata checking will depend on each data type. The Preprocessing CSC performs some or all of the following field and range checks for metadata on an ingest data type by data type basis:

- a. Field Checking. Verifies all required metadata parameters exist and that the metadata parameters use correct syntax.

ECS has defined a set of "core metadata" which will be supplied for ECS products. This core set of metadata parameters define the minimum metadata which can be supplied for an ECS product. The core metadata will also dictate a specific format and syntax for each parameter. Metadata for a specific data type will have the minimum or core set of metadata and optional metadata data type unique fields. Some data types (e.g., external ancillary data, Level 0 data, Version 0 data) may not be required to meet these core metadata requirements. Level 0 data may not be required to contain this core metadata since it will not be widely accessed by the user community. External ancillary data and V0 migration data may have existing metadata standards and heritage metadata. This does not preclude other subsystems within ECS from generating core metadata for the external ancillary data sets and V0 migration data sets.

The Preprocessing CSC will determine the required metadata parameters for a specific data type through interaction with the Data Server Subsystem. The Data Server Subsystem supplies a target metadata configuration file dictating the required parameters for the specified data type. A scenario illustrating the full metadata field checking process is included in Table 4.4-10.

- b. Range Checking. Verifies for selected metadata parameters, that parameter values lie within a specified range or that the parameters are within a set of discrete values (valid options). The purpose of range checking is not to flag values that may have relevant scientific values outside the normal range of values but to flag fields with values outside of possible range limits or not in a set of possible discrete options. Default values for some selected metadata fields may be inserted when a value is missing.

The Preprocessing CSC will rely on the Data Server Subsystem for validation of metadata files. The Preprocessing CSC will first convert the input metadata into a final PVL metadata file and then utilize data server validation services.

- c. Byte Checking: In some cases the metadata provided with a specific science data type will indicate the size of the science data packet or file. In this case, the InDataPreprocessTask Class will check that the size of the packet/file is within a certain tolerance.
- d. Time Range Checks: Time information in metadata will be checked and translated to the ECS standard format as necessary.

**Ingest Unique Metadata:** All data types will have ingest unique metadata appended to their original metadata. This ingest unique metadata consists of time of ingest and a quality flag.

#### 4.5.6 Operator Ingest Interface CSC

The Operator Interface CSC provides operations personnel with the capability, via the GUI Interface, to perform physical media ingest; to monitor the ingest history log; to monitor the status of ongoing ingest requests; to update (i.e., cancel, suspend, resume, and change priority) an ingest request and to modify Ingest configuration parameters (tunable parameter). The CSC is comprised of the GUI Interface, InGUISession, InIngestMain Window, InMediaIngest, InMediaCheckIn, InHistoryLog and InRequestController object classes. The CSC is initiated as a set of GUI screens with associated software to process GUI input when authorized operations personnel select any of the defined operator service options from the GUI screens. This CSC corresponds to the "Ingest Subsystem Administration Application" CSC described in the SDPS System Design Specification. The Operator GUI Interface contains different screens associated with each operator task. From the GUI screen, the operations personnel select the desired task to be processed and enter the information needed for building the request for the task. The GUI Interface makes sure that all the required fields are properly filled by the operations personnel. If all required fields are indeed properly filled, the GUI Interface invokes the appropriate services provided by this CSC to fulfill the task.

The InMediaIngest, InHistoryLog, and InRequestController object classes are derived from the InIngestMainWindow objects class. This means that all of these object classes inherit the data and service members from the InIngestMainWindow object class.

The InMediaIngest object class provides services to perform physical media ingest. The object class performs privilege check on both the operations personnel and the media provider. The object class interfaces with the operations personnel via the GUI Interface. A DAN (or "Media Delivery Record") must be supplied with each piece of media. The DAN describes the contents of the media.

The InMediaCheckIn object class maintains a list of tapes which the operator has "checked in" for subsequent processing. This class is provided to allow the operator to check in tapes as they arrive via mail or courier. The operator can thereafter, refer to the MediaCheckIn GUI display to determine which tapes are currently being processed and which tapes remain to be processed. (The History Log class described below, will provide the operator information on completed tape ingest request).

The InMediaCheckIn class allows the operator to utilize a bar code reader (where available) to supply the type ID or to manually type in the tape ID via a GUI screen.

The InHistoryLog object class provides services to monitor Ingest History Log (an alias for Data Receipt Log). The service allows the operations personnel to specify the search criteria for the Ingest History Log entries for viewing based on 1) ingest start/stop date and time, 2) data provider

ID, 3) data set name, 4) final request status, and 5) ingest type. The InHistoryLog object class is responsible for gathering the search criteria information input by the operations personnel via the GUI Interface and interfaces with the InRequestSummaryHeader and InRequestSummaryData object class to display the Ingest History Log. The InRequestSummaryHeader and InRequestSummaryData object class is defined in the Ingest Administration Data CSC; refer to that CSC for details.

The InRequestController object class provides service to monitor and control the ongoing ingest requests. Authorized operations personnel are allowed to view all or selective ingest requests in the system, whereas a less privileged user can view only the requests that are owned by the individual. The service allows the operations personnel to select Ingest Requests for monitoring based on: 1) data provider ID, and 2) ingest request ID, or the operator can request to monitor all requests. The object class is responsible for gathering the search criteria information provided by the operations personnel via the GUI Interface and interfaces with the InRequestProcess Header and InRequestProcessData to get the results. The object class is responsible for detecting requests where the specified expiration data/time has been exceeded. In that case the operator is alerted at the GUI screen. Refer to the Ingest Request Processing CSC for details on the two object classes.

In addition to monitoring capabilities, the InRequestController object class provides service to update an ongoing ingest request. The updates include change priority, cancel, suspend, and resume. The object class is responsible for reading in from the operations console the update specification (e.g. change priority, cancel) and the request (or requests) to update. It interfaces with the InRequestManager object class to locate the request on which the update is to be performed. Refer to Ingest Request Processing CSC for details on the InRequestManager object class.

Finally, the InRequest Controller Class allows the operator to modify data base stored parameters affecting the functioning of the Ingest system. These parameters include all of the Ingest tunable parameters discussed in Section 3.2.5.2 and the data server URs discussed in Section 3.2.4.2.

Figures 4.5-6, 4.5-7, and 4.5-8 provide a pictorial overview of the Media Ingest, the Administrative Viewing, and the Request Cancellation processing respectively. The Administrative Viewing diagram (Figure 4.5-7) describes History Log Monitoring and Request Status Monitoring GUI interfaces in a very generic fashion because of the similarity in their processing--both interface accesses the Ingest DBMS to generate a report for display based on the entries queried from the database.



#### 4.5.7 Interactive Ingest Interface CSC

The Interactive Ingest Interface CSC provides ECS science users and ECS operators with the capability to interactively request the ingest of network accessible data into ECS. In particular, this Interactive Interface allows the ECS operator to ingest Digitized Document Files produced by the Ingest Digitizing/Scanning COTS product. For such ingests, the operator will be prompted for metadata information necessary for later reproduction or viewing of the digitized/scanned product. In Release A documentation and at Release B IDR, the Interactive Ingest Interface CSC was referred to as the User Network Ingest Interface CSC.

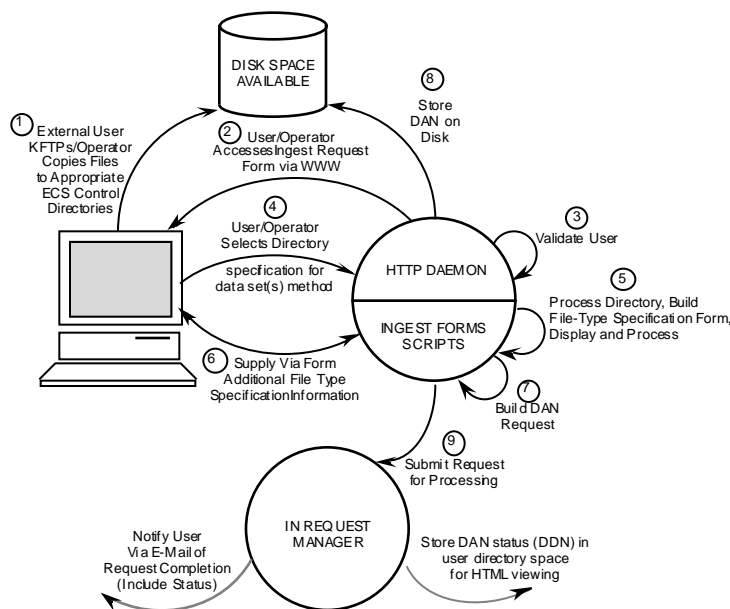
ECS science users can, via this interactive interface, request the status of their ongoing ingest request. The ECS operator can choose to view request status via this interface or via the Operator Ingest Interface. The CSC is composed of an HTML Forms Interface, the `InInteractiveIngest`, and the `InRequestProcessHeader` object classes. The CSC is initiated when the user selects ECS Ingest Interactive Interface on the World Wide Web (WWW). This CSC corresponds to the "Ingest Preparation Toolkit" CSC described in the SDPS System Design Specification.

The HTML Interface consists of HTML Forms, customized to science or document data which allows users to interactively fill in information needed for building the interactive ingest request or the status monitor request. The HTML Interface makes sure that all the required fields are properly filled by the user. If all required fields are properly filled, the HTML Interface invokes the appropriate services provided by the `InInteractiveIngest` or the `InRequestController` object class to fulfill the request. The Interactive Ingest Interface CSC provides multiple ways in which the user can specify data sets for ingest. The user can select the HTML form supported method which best suits the data set(s) to be ingested. The user can choose to: 1) specify the directory or directories where one or more data sets reside. The Ingest software will consider all files in a directory as a single data set. Note that valid ECS controlled directories are pre-established and user/datatype dependent; 2) specify a single directory where one or more data sets reside. The Ingest software will separate the files in this directory into multiple data sets using the following file naming convention; all files with the same base file name will determine a data set. (That is, all files within a data set have file names that only differ in the file extension.) It is assumed this method will be useful for ingesting documents with associated ASCII metadata files; 3) interactively specify one or more file names/locations which identify one or more data sets [interactively build a Data Availability Notice (DAN) file]; 4) specify the name and location of a DAN file containing data set file description. Note that for security purposes the DAN file must be located in an ECS controlled pre-established directory allocated to the requesting user. In all but the DAN method, the user will be prompted to supply a valid ECS data type ID for the data set(s) plus a file type for each file in the set. From the user provided information, the Interactive Ingest Interface software will create a DAN file, validating that all pertinent information has been supplied. The user can then choose to ingest the data set(s) or choose to use the created DAN file at some later time to request an ingest (e.g. as input to an operator hard media Ingest session.)

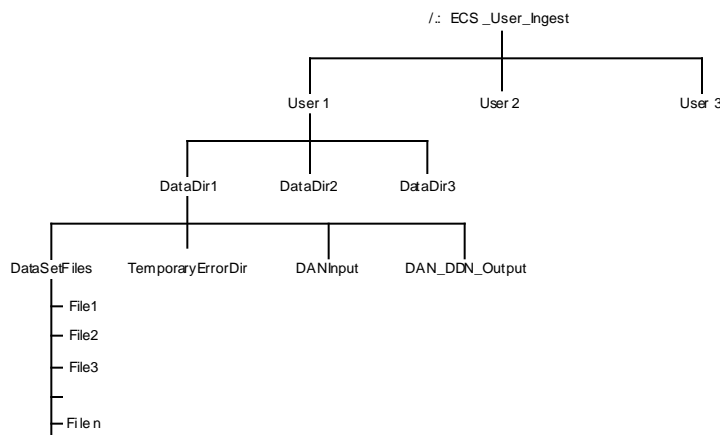
Figure 4.5-9 provides an pictorial overview of the Interactive Ingest processing when the user has chosen to specify the directory location of a collection of files making up a single data set. The diagram depicts how a user would ingest data files into the ECS system using HTML Forms. Before invoking the Ingest Form Script defined in `InInteractiveIngest` object, the fields in the Request Form will be parsed by the HTTP daemon. Upon invocation, the Ingest Form Script will package the Request Form fields into a DAN message and then send the message to ECS Ingest

(InRequestManager object) for processing. The InRequestProcessHeader object class provides services for the user to view the state of the user ongoing ingest request. The user is authorized to view only the requests that are owned/submitted by the user. This object class is defined in the IngestRequestProcessing CSC.

As Figure 4.5-9 indicates, the user is required to transfer the files (data sets or the DAN file describing the data sets) into ECS controlled directories prior to initiating the HTML Interactive Ingest Session. Figure 4.5-10 provides a sample directory structure for the controlled directories (KFTP or copy). Only valid users will have write access to these directories.



**Figure 4.5-9. Interactive Ingest Interface CSC**



**Figure 4.5-10. Sample ECS User Ingest Directory Structure**

Once the Ingest is completed, data sets which have been successfully archived will be deleted by Ingest InRequest software. If a data set failed to be archived, InRequest will rename the files into a temporary error directory. The user will be able to replace files with errors and then resubmit the Ingest Request. InRequest object will create a Data Delivery Notification (DDN) file for both successful and non-successful Ingests. The DDN will be available for viewing via the HTML Interface. The InRequest object is defined in the IngestRequestProcessing CSS.

#### **4.5.8 Ingest DBMS CSC**

The Ingest DBMS CSC is responsible for storing and providing access to Ingest Subsystem internal data. The CSC is composed of an OTS Data Base Management System (DBMS). In particular, the DBMS stores the Ingest operations data bases -- Ingest History Logs and Ingest request checkpoint state, and template information. Those data bases are configured in the Ingest Administration Data CSC, which is described in the following paragraph. This CSC corresponds to the "Subsystem Administration DBMS" CSC described in the SDPS System Design Specification.

#### **4.5.9 Ingest Administration Data CSC**

The Ingest Administration Data CSC table descriptions of administrative information for the Ingest operations data bases are maintained by the Ingest DBMS. Stored procedures used to manage the tables are also provided by this CSC. The Ingest operations data bases are described at a high level in section 4.6.1.4 and at a more detailed level in the object model in Section 4.3. The CSC interfaces with MSS to allow generation of production reports. Refer to Section 4.6.3 for a description of proposed production reports and to the MSS volume of this document for a description of associated MSS event logs.

#### **4.5.10 Peripherals CSC**

The Peripherals CSC described in the SDPS System Design Specification is entirely reused from the Peripherals CSC software described in the Data Server Subsystem volume of this document. That CSC provides common access of the Distribution and Ingest Peripheral Management HWCI (DIPHW, as described in volume 9 of this document) for ingest and distribution purposes. The Peripherals CSC is configured on the DIPHW CSCI.

#### **4.5.11 Viewing Tools CSC**

The Viewing Tools CSC described in the SDPS System Design Specification is entirely reused from the visualization and other client tools described in the Client Subsystem volume of this document. Viewing capabilities are provided only for ingested data (i.e., only for data already converted/reformatted into forms accessible by standard ECS viewing tools). Data that is not converted/reformatted into a form accessible by standard ECS viewing tools are not available for viewing or other analysis within the Ingest Subsystem. The Viewing Tools CSC is configured on the Ingest Client HWCI (ICLHW) administration workstation component or by means of x-terminal access of the Ingest Client host.

#### **4.5.12 Data Storage Software CSC**

The Data Storage Software CSC described in the SDPS System Design Specification is entirely reused from the CSCs comprising the Science Data Server (SDSRV) CSCI and Storage Resource Management (STMGT) CSCI described in the Data Server Subsystem volume of this document. Those CSCs provide for reliable storage and retrieval of the Level 0 (L0) data in the same fashion as for other data products. No additional Ingest Subsystem software is required to support L0 high-reliability data storage and subsequent access. The Data Storage Software CSC is configured on the Ingest Client HWCI (ICLHW) host component for Level 0 data ingest.

#### **4.5.13 Resource Administration CSC**

The Resource Administration Application CSC described in the SDPS System Design Specification is entirely reused from the administration CSCs comprising the Science Data Server (SDSRV) CSCI and Storage Resource Management (STMGT) CSCI described in the Data Server Subsystem volume of this document. Those CSCs provide for operations staff monitoring and control of storage components. No additional Ingest Subsystem software is required to support L0 high-reliability data storage and subsequent access. The Resource Administration CSC is configured on the Ingest Client HWCI (ICLHW) administration workstation component (or accessed by x-terminal from the Ingest Client host) for Level 0 data ingest.

#### **4.5.14 Client Interfaces CSC**

The client CSC is entirely reused from the Science Data Server (SDSRV) CSCI described in the Data Server subsystem volume of this document. That CSC provides public objects/services to perform data granule insertion. The Client CSC is configured on the Ingest Client HWCI (ICLHW) host component for Level 0 data ingest.

### **4.6 Ingest CSCI Management and Operation**

The materials in the following paragraphs discuss the management and operations of software components discussed in section 4.5.

#### **4.6.1 System Management Strategy**

The Ingest CSCI is designed to provide robust ingest services to external data providers. Specifically, the design goal of the Ingest CSCI is to always return status (successful or unsuccessful) for every received ingest request. To accomplish that goal, the Ingest CSCI follows ECS project guidelines for:

- Process startup and shutdown;
- Error detection and reporting;
- Fault tolerance and error recovery;
- Ingest operations data bases.

##### **4.6.1.1 Ingest Startup/Shutdown**

As described in the MSS volume of this document, MSS provides life-cycle services for system startup and shutdown. The Ingest subsystem fully uses those services.

At system startup, the Ingest InServer object, the InRequestManager object, and the InGranuleServer\_S object are instantiated as standalone processes. In addition, data base tables corresponding to the InRequestProcessHeader, InRequestSummaryHeader, InRequestProcessData, InRequestSummaryData, and InRequestFileInfo are set up prior to the initial system startup.

The InGranuleServer\_S process is instantiated on one or more processes as described in section 4.5.3, Ingest Request Processing.

InServer, InRequestManager, and InGranuleServer\_S are “object factories” which create objects as separate pthreads as ingest requests pass through the ingest system. Refer to section 3.2.3.1 for a detailed discussion on object factories.

When Ingest processes are started, they check for the existence of checkpointed information. If such checkpointed information is available, the Ingest process will restore the information and continue processing. Note: there is a tunable time limit after which checkpointed information will not be restored. Additional checkpointing strategy is discussed in section 4.6.1.3.

#### 4.6.1.2 Error Detection and Reporting

As described in the CSS and MSS volumes of this document, CSS and MSS jointly provide event logging services for logging and reporting errors and faults, and for browsing error/status logs. The Ingest subsystem fully uses those services. Errors detected in the processing of the InRequest object are identified in Table 4.6-1, which shows critical errors reported and the actions (including operations personnel actions) taken.

Errors/status may be reported in two error logs. MSS maintains the first log, the MSS event log. It contains errors/status of interest to operations staff to evaluate system status and to perform trend analysis. The Ingest subsystem maintains the second log, the Ingest event log. The Ingest event log contains selected errors/status from the MSS event log (for context) plus highly-detailed debug events. Software maintenance personnel use the Ingest event log to diagnose system and software problems in response to trouble tickets.

**Table 4.6-1. Ingest Subsystem Error Categories (1 of 3)**

Error Category	Actions to Be Taken
Metadata Validation Failure	In general, log errors to the event log and return status to the external data provider. Operations staff evaluate errors off-line and request re-ingest after corrections  Dependent on preprocessing templates (instituting DAAC policy), continue with preprocessing and insertion of data into the data server subsystem in a special data type category, "INCOMPLETE" (with metadata quality set to "UNVALIDATED"). Flag Metadata indicating that the metadata was not validated. Operator action and source facility notification required. Log errors to the event log and return status to the external data provider. Operations staff evaluate errors off-line and request re-ingest as necessary
Target/Source MCF Mismatch Failure	Same as above. Operations staff are alerted that preprocessing templates are out-of-synch. Operations staff develop a trouble ticket to correct the mismatch

**Table 4.6-1. Ingest Subsystem Error Categories (2 of 3)**

Error Category	Actions to Be Taken
Data Type Policy Failure: No Metadata	Log errors to the event log and return status to the external data provider. Operations staff evaluate errors off-line and request re-ingest after corrections
Data Type Policy Failure: Incomplete set of Files, Metadata exists	<p>In general, log errors to the event log and return status to the external data provider. Operations staff evaluate errors off-line and request re-ingest after corrections</p> <p>Dependent on preprocessing templates (instituting DAAC policy), continue with preprocessing and insertion of data into the data server subsystem in a special data type category, "INCOMPLETE" (with metadata quality set to "UNVALIDATED"). Flag Metadata indicating that the metadata was not validated. Operator action and source facility notification required. Log errors to the event log and return status to the external data provider. Operations staff evaluate errors off-line and request re-ingest as necessary</p>
Data Type Policy Failure: Unknown Data Type	Log errors to the event log and return status to the external data provider. Operations staff evaluate errors off-line and communicate with external data provider. (Note: all data sets to be ingested must be pre-authorized by DAAC personnel. Ingest and Data Server setup is required before ingest and archive can occur)
Data Type Policy Failure: Unknown File Type.	Same as above
File Type Failure	<p>In general, log errors to the event log and return status to the external data provider. Operations staff evaluate errors off-line and request re-ingest after corrections</p> <p>Dependent on preprocessing templates (instituting DAAC policy), continue with preprocessing and insertion of data into the data server subsystem in a special data type category, "INCOMPLETE" (with metadata quality set to "UNVALIDATED"). Flag Metadata indicating that the metadata was not validated. Operator action and source facility notification required. Log errors to the event log and return status to the external data provider. Operations staff evaluate errors off-line and request re-ingest as necessary</p> <p>Operations staff are alerted that preprocessing templates are out-of-synch. Operations staff develop a trouble ticket to correct the mismatch</p>
Unable to archive data	Internal Data Server fault. Log errors to the event log and return status to the external data provider. Report alert to operations staff. Operations staff evaluate errors off-line and request re-ingest as necessary
Unable to read peripheral media	Internal Data Server fault. Log errors to the event log and return status to the external data provider. Report alert to operations staff. Operations staff evaluate errors off-line and request re-ingest as necessary
Unable to transmit data ingest coordination messages	After a system-tunable number of retries, log errors to the event log. Report alert to operations staff. Operations staff evaluate errors off-line to evaluate and correct communications network problems

**Table 4.6-1. Ingest Subsystem Error Categories (3 of 3)**

Error Category	Actions to Be Taken
Unable to transfer data to be ingested	After a system-tunable number of retries, log errors to the event log and return status to the external data provider. Report alert to operations staff. Operations staff evaluate errors off-line to evaluate and correct communications network problems
Unable to allocate disk space	Allocation request is queued by Data Server and satisfied when disk spare becomes available. Allocation requests exceeding DAAC specific limits will be rejected by the STMGTS CSCI. Log failures to the event log and return status to external data providers. Report alert to operations staff. Operations staff evaluate errors off -line and request re-ingest as necessary
Unable to set up external data provider session	Limit exceeded for allowable number of external data provider sessions. Log errors to the event log and return status to the external data provider, indicating that the session connection should be re-attempted later. (Note: based on the modeled transaction load, this error condition is expected to occur very rarely, if at all)

#### **4.6.1.3 Fault Tolerance and Error Recovery**

Once an ingest request is accepted from an External Data Provider, the ECS policy is to complete request processing and return status (successful or unsuccessful) to the External Data Provider. Therefore, upon creation of the InServer, InSession, InRequestManager, InRequest, and InGranuleAsync\_S objects, their critical attributes are checkpointed to a COTS data base. After a process or system failure, the checkpointed attributes are automatically restored to the last checkpointed state and processing continues. Note: there is a tunable time limit after which checkpointed information will not be restored.

Data bases for InServer/InSession, and InRequestManager/InRequest are checkpointed only when a session or request, respectively, changes its fundamental state. Any change to state that does not invoke checkpointing is defined as a "substate."

The InServer object has two fundamental states--"Session Created" and "Session Deleted." The InServer object is checkpointed after each InSession object is created (storing session ID) and after each InSession object is deleted (deleting session ID). The InServer object has two substates--"Active" (at object creation) and "Inactive" (at object deletion) that are not checkpointed.

The InSession object has three fundamental states--"Active", "Request Created", and "Request Deleted." The InSession object is checkpointed immediately after it is created (storing information about the connection with the external data provider); after a InRequest object is created (storing the request ID); and immediately after the InRequest object is deleted (deleting request ID).

The InRequest object has seven fundamental states -- "Active", "Suspended", "Cancelling", "Cancelled", and "Resuming", "Complete" and "Complete with Errors". The InRequest object is checkpointed to "Active" when created by InRequestManager. The InRequest object is subsequently checkpointed to "Suspended", "Cancelling", or "Resuming" when an operator command is received to suspend, cancel, or resume, respectively. Once all granules within the request have been suspended, cancelled, or resumed, the InRequest object is checkpointed to "Suspended", "Cancelled", or "Active", respectively. Note: while a request is suspending,

cancelling, or resuming, all additional operator state change requests will be refused. At completion the request is checkpointed as “Complete” or “Complete with Error” as appropriate.

InGranuleAsync\_S objects have seven states -- “Active”, “Transferred” “Preprocessed”, “Inserted”, “Transfer Failed”, “Preprocess Failed” and “Insert Failed”. InGranuleAsync\_S object in the Active State have additional checkpointing at granule file level -- “File Not Transferred”, “File Transfer Failure”, and “File Transferred”.

The above checkpointing allows for the recovery of requests after a process or system failure. Granules not completing ingest prior to the failure will be resumed following last successful checkpoint. Ingest will checkpoint when a granule recovery is attempted. If a process or system failure occurs again during the recovery, Ingest will not attempt to recover the granule a second time.

Failure scenarios with recovery methods:

- 1) Failure of the InServer object. This process is restarted by MSS Process Framework services as a Unix standalone process. The list of active session IDs is restored from its checkpointed state. Corresponding InSession processes are restarted if they were disabled. Prior to restart, no External Data Providers may set up new connections (connection requests are rejected). Prior to restart, InSession objects that have completed all interactions with an External Data Provider are blocked from terminating.
- 2) Failure of an InSession object. The InServer process detects the failure via a system signal. InServer creates a new InSession object that communicates with the External Data Provider and with the submitted InRequest based on the checkpointed list of request IDs (OODCE OID). Prior to restart, the External Data Provider application may time out. The InRequest object that has subsequently completed is blocked until the InSession object is restored. No other ingest sessions that have not failed are affected.
- 3) Loss of the data base tables used for checkpointing. The data base management system (DBMS) automatically checkpoints transactions to allow restorations of table information. In the event of a hard disk failure/DBMS failure, the InRequest object detects the failure, and reports a request failure condition to the InSession object. Since RAID-1 (mirrored disks) are used, this failure should occur very rarely.
- 4) Failure of InRequestManager. All associated InRequest pthreads are terminated. The InRequestManager process is restarted as a Unix standalone process. The identifiers of InRequest objects are restored from checkpointed information. The InRequestManager recreates the InRequest objects and they continue processing from their latest checkpoint (see the next scenario).
- 5) Failure of individual InRequest pthreads. The InRequestManager process detects the failure and creates a new InRequest object that communicates with the originating InSession object based on the checkpointed request ID (OODCE OID). InGranuleServer\_S and InGranuleAsync\_S objects will not be affected. The InRequest object will recreate (using checkpointed SRF provided object URs) corresponding InGranuleServer\_C object for active InGranuleAsync\_S objects. Messages from server objects are queued during the time the client objects are down and are processed once the object is re-created. No other ingest requests that have not failed are affected.

- 6) Failure of an InGranuleServer process. All associated InGranuleAsync\_S objects (pthreads) are terminated. The InGranuleServer\_S process is restarted as a Unix standalone process. InGranuleServer process recreates all InGranuleAsync\_S objects and based on checkpoint information, continues processing from point of failure.
- 7) Failure of individual InGranuleAsync\_S pthread. The InGranuleServer\_S object will recreate the failed InGranuleAsync\_S pthread and restart processing from point of failure.
- 8) Failure of the processor on which an Ingest process is running. In general, operations personnel restart the processor. Restart of individual processes is handled as a combination of one or more of the above process restarts. If the processor is disabled, the disablement is detected by MSS SNMP services and, where available, a backup processor is restarted. The backup processor has full access to the data base tables used for checkpointing and has an identical network address to that of the primary processor. Again, restart of individual processes is handled as described above. Note: in the Ingest subsystem a backup processor is generally provided only in the case of Level 0 data ingest, where a separate set of Ingest Client processors is provided for high availability.
- 9) Failure of the External Data Provider application. After a given number of retries to transmit the Data Delivery Notice (DDN), operations staff are notified by means of an alert message. The Data Delivery Acknowledgement (DDA) is simulated for InSession, which exits normally. The DAAC operations staff will coordinate with the External Data Provider operations staff to diagnose the failure.
- 10) Network failure for the connection with the External Data Provider application. Same as 9).

#### **4.6.1.4 Ingest Operations Data Bases**

The Ingest CSCI maintains three types of operations data bases:

- Ingest checkpoint history logs
- Ingest template information
- Ingest configuration information

Ingest history logs contain detailed and summary information about ingest request status. The two types of information are complementary and fulfill different operational requirements. Both types of information are stored in a COTS data base management system.

The detailed information is used to restore system state after a process or system failure as described above. Selected information (e.g., active request IDs, request state) is available for display of status on ongoing requests. The detailed information is deleted when processing for a request is complete. In the object class descriptions in Section 4.3, classes InRequestProcess Data, In RequestProcessHeader, and InRequestFileInfo contain the detailed information.

The summary information is used to maintain longer-term summary statistics on ingest processing. Selected information (e.g., request IDs, data volume ingested, number of data granules successfully/unsuccessfully ingested) is available for display. The summary information is updated during ingest request processing and maintained after request processing is complete. Summary information is deleted periodically based on DAAC operational policy. In the object

class description in Section 4.3, classes `InRequestSummaryHeader` and `InRequestSummaryData` contain the summary information.

Ingest template information is stored by sustaining engineering personnel to implement data-specific functions. Sustaining engineering personnel are responsible for entering and updating templates that guide ingest data processing. Separate templates are implemented for data type information, file type information, and metadata configuration information. In the object class descriptions in Section 4.3, classes `InDataTypeTemplate`, `InFileTypeTemplate`, and `InSourceMCF` contain template information.

Ingest configuration information includes parameters for ingest thresholds (e.g., maximum number of requests to concurrently process maximum data volume to concurrently process, number of data transfer retries) allow operation personnel to throttle ingest processing and the flow of data into site Data Servers. `InExternalDataProviderInfo` controls the registering of threshold parameters with Managed Process Framework as the Ingest `InRequestManager` Process is initialized. Operations personnel are responsible for updating these parameters as necessary to fine tune the Ingest system performance. Refer to Section 3.2.4.7, Performance and Accountability Reporting, for more details.

Ingest configuration parameters also include the tunable parameters referenced in Section 3.2.5.2 (e.g., number of data retries, time limit completed request remain on the operator monitor screens). The `InExternalData ProviderInfo` and `InSystemInfo` objects contain the tunable parameters.

Operations interfaces to the operations data bases are discussed in section 4.6.2. Additional uses of the operations data bases are discussed in Section 4.6.3. Sustaining engineer interfaces to the operations data bases are discussed in Section 4.6.4.

## **4.6.2 Operator Interfaces**

The Ingest GUI interface provides a collection of GUI components through which privileged users (e.g., operator) can access services in ECS Ingest. The Ingest GUI interface is categorized into three groups: Administration, Media Ingest, and Interactive Ingest.

The Administration interface provides operations personnel with the capability to monitor and control all of the Ingest activities in the system. The interface is composed of three GUI screens.

The first Administration interface is Request Control, which allows operations personnel to monitor and control ongoing request processing states based on the specified criteria. Operations personnel can use this same GUI interface to delete, cancel, suspend or resume requests. The second Administration interface is Ingest History Log Viewing. The History Log provides a summary of Ingest activities that have happened. The log entries can be queried through a selection of criteria.

The Media Ingest interface allows operations personnel to perform physical media Ingest. The types of media supported in physical media ingest in Release A are 4mm tape, and 8mm tape. Release B additional media types (e.g., for ASTER LIA/LIB ingest) are TBD.

The Interactive Ingest interface allows privileged users to electronically ingest data into the ECS system. The Interactive Ingest Interface provides multiple ways in which the user can specify data sets for ingest into ECS. Refer to Section 4.5.7, Interactive Ingest Interface CSC for details. Science data and document data are expected to be ingested via this interface. In addition to the

ingesting service, the Interactive Ingest Interface provides users the capability to view the status of submitted ingest requests. This service allows users to find out the processing state of a previously-submitted request.

The third Administration GUI interface is the Ingest Tool interface. It provides operator access to the UR update tool referenced in section 3.2.4.2, Universal Reference/Advertising and a means to update Ingest Tunable parameters (e.g., Ingest Polling interval, Ingest request priority). See Section 3.2.5.2, Tunable Parameters for a complete list of Ingest Tunable parameters. The UR update tool should be executed only at initial Ingest System setup and when Data Server URs have changed. It is anticipated that URs will change infrequently. Operations should be notified in advance of a UR change so that the UR update tool can be executed. If prior notification is not received, the operator will be notified by the Ingest software that URs are invalid and that the UR update tool should be run.

In addition to the Ingest Administration, Media Interactive, and Operator Interfaces, the operator can, via the MSS provided interface, view and update Ingest thresholds parameter values. These thresholds consist of 1) the maximum allowed Ingest Request to be processed concurrently, 2) the number of transfer retry attempts when network failure occurs, 3) the Polling timer indicating how long to wait before starting the Ingest Polling, and 4) the maximum allowed data volume to be processed concurrently.

All of the Ingest GUI interfaces will be implemented based on the "ECS User Interface Style Guide" document. The document defines standards for the ECS User Interface design and implementation. The intent of using the guide for GUI development is to help ensure that ECS has a consistent and common look and feel user interface system wide.

The Administrative and the Media Ingest interfaces will be developed in X-Windows/Motif. The Interactive Ingest service will be developed in HTML.

DID 605, Operation Scenario for the ECS project (605-CD-002-001) describes and illustrates all of the Ingest GUI screens for Administration and Media Ingest.

Table 4.6-2 provides a summary of all Ingest HTML forms for the Interactive Ingest Interface.

### **4.6.3 Ingest Production Reports**

In addition to ad hoc ongoing request status displays discussed above, the Ingest subsystem provides the standard reports described in Table 4.6-3.

**Table 4.6-2. Ingest HTML Forms**

Method of Ingest	FORM(S)	Input	Output
All	InteractiveIngestMain	-UserName Method of Ingest or Status Request selection Submit Request/Build DAN only option selection	Appropriate Method Form
Single Data Set Directory	- SingleDataSetDirectory  -FileTypeForm  -	- Data Type Directory - File Type for each displayed file name	- Validation Status Form Next form - Validation Status Form DAN ID RequestID (if submit option is selected) Next form (Interactive IngestMain)
Multiple Data Set Directory	-MultiDataSetDirectory  -FileTypeForm  -Repeat FileTypeForm for every granule in directory	- Data Type Directory - File Type for each displayed file name ....	- Validation Status Form Next form - Validation Status Form  ....  DAN ID RequestID (if submit option is selected) Next form (Interactive IngestMain)
Interactively Build DAN	- DANHeader  - DataTypeForm  - FileEntryTypeForm	- OriginatingSystem SequenceNumber ExpirationTime - Data Type of data set  - File Location File Name File Type File Size (or select Determine File Size option	- Validation Status Form Next form  - Validation Status Form Next form - Validation Status Form DAN File Name RequestID (if submit option is selected) Next form (Interactive IngestMain)
Supply DAN File Name	DANFileName	- DAN File Location DAN File Name	- Validation Status Form RequestID (if submit option is selected) Next form (Interactive IngestMain)
Request Status of on going request	StatusForm	- RequestID or UserID	- Validation Status Form Request(s) Status

**Table 4.6-3. Standard Ingest Production Reports**

Report Type	Report Description	Intended Audience
Ingest History Report (summary and detailed versions)	The report gives a detailed account of all ingest requests processed during a reporting period, as well as summary statistics. The report supplies operations staff with a log and summary view of ingest request completion performance (e.g., maximum, minimum, average volume of data ingested). The report is generated on a daily, weekly, and monthly summary basis, and as an ad hoc report	Ingest Technician Resource Planner Performance Analyst Operations Supervisor DAAC Assistant DAAC Manager

#### **4.6.4. Sustaining Engineering Interface to Data Processing Templates**

Sustaining engineers can update the data/file type policy and source/target format information maintained in Ingest Operations Data Base via the Sybase provided Interactive SQL (ISQL) supported by Ingest provided stored procedures. The engineer will enter updates to the Data Type Template table, the File Type Template and the Source MCF data base tables to add or modify a data type in the Ingest Data Processing Template Tables. The DataType Template, FileType Template and Source MCF Sybase data base tables drive the Ingest metadata extraction and data conversion and validation software.

The sustaining engineer will be able, via Ingest provided stored procedures, to obtain formatted reports on existing data type, file type and source MCF template data base table content. Ingest will also provide store procedures to support the addition, deletion, or modification of a data type and corresponding file type and source MCF template information to the data base. Table 4.6-4 describes Ingest provided stored procedures. It is expected that the sustaining engineer will modify a test version of the operational data base using the provided stored procedure. The modification will be validated by the Integration and Test personnel running Ingest in a test mode. Once tested, the sustaining engineer will, via Sybase, create a backup of the modified template tables. The template table backup will then be used to replace the operational data base template tables on the operational Ingest data base system.

The sustaining engineer can execute the stored procedure in batch mode. Ingest will provide a “sample” batch execute file for each stored procedure. The batch execution file will provide detailed descriptions of required stored procedure input parameters. The engineer will be required to edit the batch execution input file to supply values for each input parameter. Once modified, the engineer will submit the batch file, via Sybase ISQL, to Sybase for processing. The stored procedure resident on the Sybase server will be executed. Ingest stored procedures and table triggers will provide appropriate consistency, completeness and range checking on all data base insertions, deletions and modifications.

DID 611, Operator’s Manual, will contain detailed directions on adding, deleting, or modifying an Ingest Data Processing Template data type.

**Table 4.6-4. Data Processing Template Stored Procedures**

Stored Procedures	Function
DataTypeReport (for all or a single specified data type)	Produces formatted file or hardcopy report on Data Type Template information (consisting of data type ID and associated Data Server and Ingest granule server URs) . The report can include information on a single specified data type or information for all data types defined in the Ingest data base
TemplateInformationReport (for all or a single specified data type)	Produces formatted file or hardcopy report on Data Processing Template tables. The report can contain information on a single data type or information for all data types defined in the Ingest data base
AddANewDataType	Adds specified data type to the Data Type Template data base. Validates that the data type does not already exist in the data base
AddANewFileTypeRow	Adds for a specified data type, all provided file type information. Verifies specified data type exists and validates provided information for type, range and completeness
AddANewSourceMCFRow	Adds for a specified data type, all provided SourceMCF information. Verifies specified data type exists and validates provided information for type, range and completeness
DeleteADatatype	Deletes specified data type from Data Type Template Table as well as corresponding entries in FileType Template and Source MCF Template Tables
DeleteAFileTypeRow	Deletes specified SourceMCF row (based on FileType/DataType composite key) from File Type Template
DeleteASourceMCFRow	Deletes specified SourceMCF row (based on SourceMCF/TargetParameter composite key) from SourceMCF Template
ModifyDataTypeUR	Modifies data type UR for given data type ID key and UR Type (Data Server or Ingest granule server)
ModifySourceParameter, ModityCSDT, ModiryFiledLocationOffset, ModityFieldLength	Modifies appropriate field in SouceMCF Template for given SourceMCF /Target Parameter composite key. Verifies type and range
ModifyMaximum, ModifyMetadataSpecialization, ModifyMetadataTarketName, ModifyMinimum, ModifyRequeiredFlag, ModifyScienceSpecialization, ModifyScienceTargetName, ModifySourceMCF, ModifyArchivalFlag, ModifyMetaDataStart, ModifyMetaDataEnd, ModifyLineDelimiter, ModifyStringDelimiter, ModifyPVSeparator, ModifyFileClase	Modifies appropriate field in FileType Template for given FileType/DataType Parameter composite key. Verifies type and range

## 5. ICLHW - Ingest Client HWCI

### 5.1 Introduction

Within ECS the Ingest Subsystem is responsible for multiple functions, including user data ingest, hard media ingest, and the monitoring of ingest status. The principal area of ICLHW responsibility in the SDPS architecture is the ingest and storage of Level 0 (L0) and other data sets for which a highly-reliable interface is required. The criticality of the timely and reliable storage of L0 data sets demands that a separate high reliability, high availability data server instantiation be dedicated to L0 data ingest. Data which has been ingested will be made available to the Processing Subsystem for the generation of higher level products. The daily volume of L0 data to be ingested at each DAAC is taken from the 2/7/96 ECS Technical Baseline, and is shown in Table 5.1-1. The Data Server volume of this document indicates the non-L0 data products which will be ingested directly into hardware provided by the Data Server Subsystem utilizing ingest client software. Data products to be ingested by Ingest Subsystem hardware include:

EDC: Landsat-7 L0R

GSFC: MODIS L0, COLOR L0

JPL: SeaWinds L0, DFA L0, MR L0

LaRC: CERES L0, TRMM H/K, MISR L0, MOPITT L0, SAGE III L0, ACRIM L0

Note: The non-L0 Landsat-7 L0R data is supplied to ICLHW due to the short duration that the L0R data is stored within the Landsat-7 Processing System (LPS).

The DAAC unique volumes of this document describe the configurations required to support the listed instruments. Sizing of Ingest Subsystem resources is sufficient to accommodate the listed volumes, as described in the following paragraphs. The daily volumes to be ingested change at each ECS release, generally increasing as additional missions are supported. The Ingest Subsystem design is therefore scalable to support the growing daily and total data volumes.

**Table 5.1-1. Daily L0 Ingest Volumes**

Daily L0 data ingest by mission by DAAC (GB/day)	TRMM	Landsat-7	AM-1	FOO/COLOR	SAGE III	ADEOS II	ALT RADAR	ACRIMSAT
ASF								
EDC		139.4*						
GSFC			70.3	3.75				0.05
JPL						0.12	0.15	
LaRC	0.25		47.5		0.125			
NSIDC								
ORNL								

\* Landsat-7 L0R data is ingested through Ingest Client Host servers, stored temporarily in Ingest Working Storage, and transferred to the permanent Data Server archive.

Functionality similar to that provided by the ingest client hosts is responsible for the ingest of data types other than L0 in the Data Server Subsystem. Ingest client software capabilities are mapped to data server components for support of the ingest of data types that are stored permanently within the data server. The ingest and archiving of these other data types is discussed in the Data Server Subsystem description volume of this document.

### 5.1.1 HWCI Design Drivers

The Ingest Subsystem hardware components consist of the client host servers, working storage, and L0 archive repository. The rationale for the separation of L0 data ingest functions from other data server functionality is discussed in Section 3.2.2. The data rates and volumes to be supported at each of the DAAC sites vary, but the basic Ingest Subsystem configuration is consistent at all sites and is described in the following section. The HWCI for the Ingest Subsystem consists solely of the Ingest Client HWCI (ICLHW), which is comprised of the ingest client hosts required for ingest management, control, monitoring, and processing of ingested data. The ICLHW also contains working storage and archive repository components similar to those of the data server Working Storage HWCI (WKSHW) and Data Repository HWCI (DRPHW). Those components utilized for the ICLHW are specialized for use in the Ingest Subsystem due to the unique ingest reliability, maintainability, and availability requirements as discussed in Section 3.2.2.1.

The client hosts manage the transfer of data into, out of, and within the Ingest Subsystem. The loading on the client hosts is principally in two areas: I/O loading associated with the ingest of Level 0 data from external sources and CPU loading required to perform basic ingest data checks, metadata validation, and metadata extraction. Additional functions to be performed include logging, status, and reporting activities, coordination of data transfers between working storage and the ingest L0 archive, maintaining a database of all data contained within the Ingest Subsystem, and servicing queries and retrievals on the archived L0 data. The principal factors affecting sizing of the client hosts are summarized in Table 5.1-2.

**Table 5.1-2. Ingest Client Sizing Factors**

Sizing factors	Driving requirements
Ingest client I/O	Receipt of L0 data from multiple external sources Servicing L0 archive queries Support of CPU loading factors
CPU loading	Data reformatting Support of ingest I/O loads Ingest data server database maintenance Ingest data checking Metadata validation and extraction

#### 5.1.1.1 Key Trades and Analysis

One trade analysis specific to the Ingest Subsystem hardware configuration was performed prior to the ECS Release A CDR. The ECS Ingest Subsystem Topology Analysis, ECS Technical Paper 440-TP-014-001, analyzes and documents the configuration of the Ingest Subsystem based on the results of the ECS Ingest Subsystem Design Analysis delivered at Release A PDR. The topology analysis focuses on factors that affect the sizing requirements of Ingest Subsystem components, as

well as the capability of the configuration to meet reliability and availability requirements. Methods for assuring acceptable system RMA, as well as the flexibility to evolve the ingest configuration as mission requirements change are discussed as key Ingest Subsystem design drivers.

The principal output of the study is the sizing of Ingest Subsystem components at each of the Release A DAACs. The component sizing was accomplished using a combination of a paper analysis of storage required to satisfy daily and annual ingest volumes, plus the development of a queuing model showing several system loading parameters at each stage of the ingest process. An analysis of the Release B data volumes and required component capabilities was subsequently performed. An additional result of this study is that the implementation of the Ingest Subsystem architecture presented at IDR will satisfy the RMA and data ingest requirements based on an analysis of key Ingest Subsystem drivers in Section 3.2. The implementation of this architecture utilizing redundant ingest client hosts in a prime/backup configuration, as well as high reliability RAID storage devices will provide the system availability and scalability necessary to support the reliable ingest of data. Outputs from this trade study have been incorporated into the design information presented in this document.

#### **5.1.1.2 Ingest HWCI Sizing and Performance Analysis**

The sizing of Ingest Subsystem hardware both from a system level and a component level is based on the 2/7/96 version of the ECS Technical Baseline. Among the information included in the baseline is data by instrument, average daily data volume by level, and data destination. The average expected daily and annual data volumes at each site were calculated from this information and used to determine the required ingest hardware capabilities. Ingest client hosts are sized to accommodate the required ingest volumes as well as I/O and CPU capabilities to support internal data transfers associated with metadata validation and extraction, L0 archive data retrieval, and transfer of data to the Data Server or Processing Subsystems. Working storage disks are sized to accommodate the above functions, as well as provide contingency space for the transfer of more than one day's worth of data within a 24-hour period. Working storage space is also effectively increased by the use of a separate L0 repository, to which ingested data is written as soon as required metadata extraction and validation is performed. The ingest L0 archive is sized to store one year's worth of L0 data, with sufficient storage and I/O capabilities to support anticipated archive write and read loads. In general, L0 data stored for more than one year (plus or minus a delta time, dependent on data type) is discarded from the repository. For L0 data without a LIA counterpart (e.g., MOPITT) the L0 data is stored permanently in the repository. Since high RMA is a driver for the Ingest Subsystem, all critical components also include some type of sparing or redundancy to ensure that availability requirements are met.

An Ingest Queuing Model was developed to assist in the sizing of Ingest Subsystem components, and is a very high level look at the data flows in the Ingest subsystem. This analysis is valid for systems where each queue has only one server, and the inter-arrival times and service times have exponential probability densities. In addition, the results are only valid for the steady-state conditions, where the probability of finding the system in a given state does not change with time.

Model output is dependent on a series of model input parameters that may be varied depending on characteristics of the data to be ingested, processed, and stored, as well as network and Ingest Subsystem component capabilities and performance. Parameters such as server CPU and I/O

performance, disk I/O, network performance, and the number of operations/byte associated with each server process (e.g., capture, format, archive, distribute) may all be varied to analyze the sensitivity of changes in data flows and system architecture. The load presented by each flow in pkts/sec is a function of the number of bits/sec input from the previous process and the mean size of packet/data set that this process expects. A "Source" item sends packets whose size and rate depend on the network technology chosen. FDDI is baselined for the Ingest Subsystem connection at all sites, with an assumed network efficiency of 60% from the maximum 100 Mbps clock rate. Refer to the Release B CSMS Communications Subsystem Design Specification (305-CD-028-002) for more details on the DAAC network architectures. A "capture" item includes server functions that receive data from the network or hard media device. Capture rate is a function of the input bandwidth and server I/O capabilities. The model includes statistics on the writing of the ingested data to working storage and the writing of data back to the server to perform the "format" operation. Both read and write operations have associated transfer rate and access time estimates for each data transfer. Conservative estimates of 2 MB/sec read and write rates are used based on results from Data Server prototyping efforts. Writing of data to the L0 archive repository involves the capabilities of the working storage disks, server I/O, and repository devices. Finally, the number of copies of data read from the archive and sent to a data sink (e.g., Processing Subsystem) may be varied to determine the additional load of reprocessing on Ingest Subsystem resources.

Competitive procurement restrictions constrain this paper to identification of the class of component and general performance characteristics used in the analysis, rather than actual candidate components. The client host CPUs are low to mid-level Symmetric Multi-Processing (SMP) 32-bit machines, capable of supporting multiple network (FDDI) and direct-connect (SCSI II) devices. Working storage devices are RAID 5 units generally with a minimum of 2.0 days worth of space allocated to ingest working storage required to support the functions of acquiring, processing, validation, and archiving L0 data. This volume of working storage allows for one day's worth of L0 data to be staged for processing, an additional day's worth available for subsequent ingest, and an additional 25% available to service additional Ingest Subsystem needs (e.g., L0 archive retrieval support, pre-processing, quality checking). Additional magnetic disk resources are supplied within the Ingest Subsystem to support items such as client host operating system and application software and L0 archive database directory information. RAID functions as the L0 repository at the smaller DAAC sites (e.g., JPL), while a robotics and tape-based repository is implemented at the sites with greater storage requirements (GSFC, LaRC). The Ingest Subsystem implementation at EDC for Landsat-7 L0R data includes only RAID working storage. The L0R archive repository is supplied by the Data Server subsystem.

Analysis of the ingest queuing model developed for CDR confirms the results of the paper analysis conducted prior to IDR supporting the ability of the Release A Ingest Subsystem configurations to support Level 0 data ingest requirements. Additional subsystem capacity is available for contingency (e.g., ingest of more than one day's worth of data in one day or resolving difficulties encountered during reformatting or metadata validation activities), Version 0 data ingest, and subsystem testing requirements. The DAAC-specific volume presents additional Ingest Subsystem hardware sizing detail and rationale.

### 5.1.1.3 Scalability, Evolvability, and Migration

Ingest Subsystem hardware must be easily scalable to support both different subsystem capabilities at each of the DAAC sites as well as changing data ingest requirements over the life of the EOS program. The architecture accommodates the required scalability through several different mechanisms within the Ingest Subsystem components as described below.

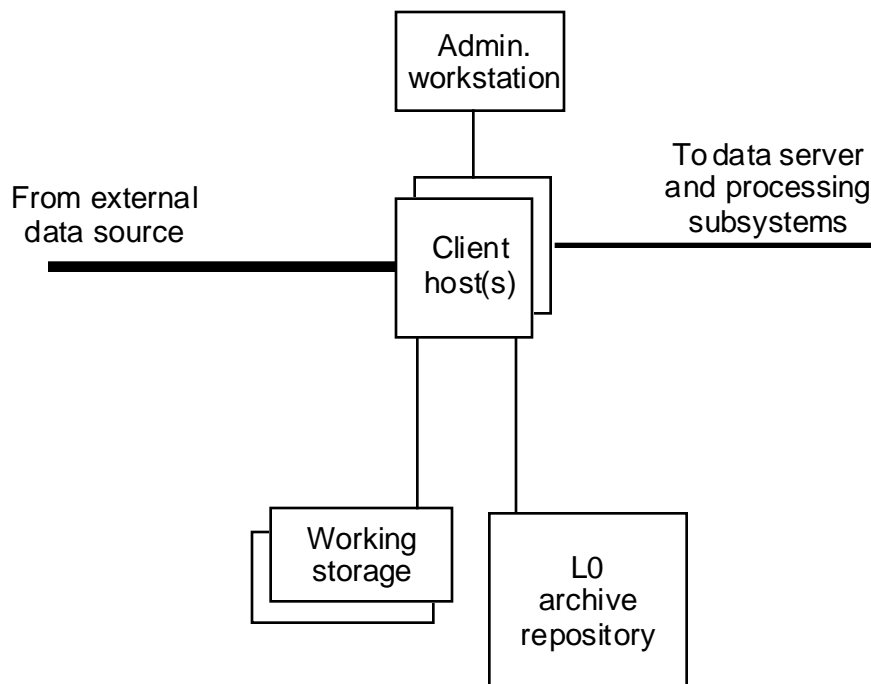
**Ingest Client Hosts** - Client host resources may be increased in two ways as support for the number and complexity of ingest clients increases. The first is through the addition of CPU, internal memory, and I/O capabilities within a given client host platform or family. The second is the addition of additional client host platforms to support additional interfaces and data providers as the ingest client load increases. Client hosts can be added virtually without limit as network attached storage devices are incorporated into the architecture. Machine upgrade or exchange for a larger class machine may be necessary where processing and I/O demands for a single interface increase significantly.

**Working Storage** - Local RAID disk banks can be added as working storage needs increase. The limiting factor becomes the number of client hosts that can be directly connected to a particular disk bank. Later ECS releases may also use network attached storage to facilitate the addition of disk space. Disks can be added to the network connections, and when the networks begin to become saturated, segmentation and subnetting techniques can improve performance.

**L0 Archive Repository** - The L0 archive repository is scalable scalability much like the client hosts and working storage components in that additional media drives within a repository or additional repositories may be added as required storage volumes increase. The ability to use multiple file servers and the mixing of storage solutions like FSMS based systems in the architecture lends itself to easy scaling. Adding additional file servers can enhance both file handling bandwidth and processing capabilities. The flat common access nature of the robotics based tape repositories allows the adding of file servers to gain access to the data without burdening existing file servers.

### 5.1.2 HWCI Structure

The Ingest HWCI consists of all of the hardware contained within the Ingest Subsystem. Client host computers control the flow of data into, out of, and within the Ingest Subsystem. Working storage magnetic disks provide temporary storage for ingested data. The L0 archive repository contains the devices required at each site for storage of Level 0 data for a period of one year (plus or minus a delta time, dependent on data type). These are the principal components which comprise the Ingest HWCI, and are described in greater detail in Section 5.1.2.2. Other peripheral devices such as administrative terminals or workstations are required for Ingest Subsystem and overall SDPS control, but are not strictly part of this HWCI. Note, however, that the Ingest instantiation of WKSHW is “highly-reliable.” Namely, redundant processors and RAID are included to ensure 0.999 availability and 15-minute switchover time. Figure 5.1-1 shows the generic Release B Ingest HWCI configuration. DAAC-specific instantiations of this configuration are contained in the DAAC-specific configuration volume.



**Figure 5.1-1. Ingest HWCI Block Diagram**

### 5.1.2.1 Ingest HWCI Connectivity

The ingest clients support several different classes of interfaces. These interfaces support ingest of data, insertion of the data into the ingest L0 archive repository, and control and monitoring of Ingest Subsystem resources. The interface classes are summarized in Table 5.1-3.

**Table 5.1-3. Ingest Client Interfaces**

Ingest Client I/F	I/F Type	I/F Instantiation Examples
External data ingest	Network	FDDI
Ingest peripherals (hard media ingest)	Network/bus	SCSI II
Ingest working storage	Network/bus	SCSI II
Ingest L0 archive repository	Network/bus	SCSI II
Client host control/monitoring	Network	FDDI, Ethernet
Ingest Subsystem control/monitoring	Network	FDDI, Ethernet

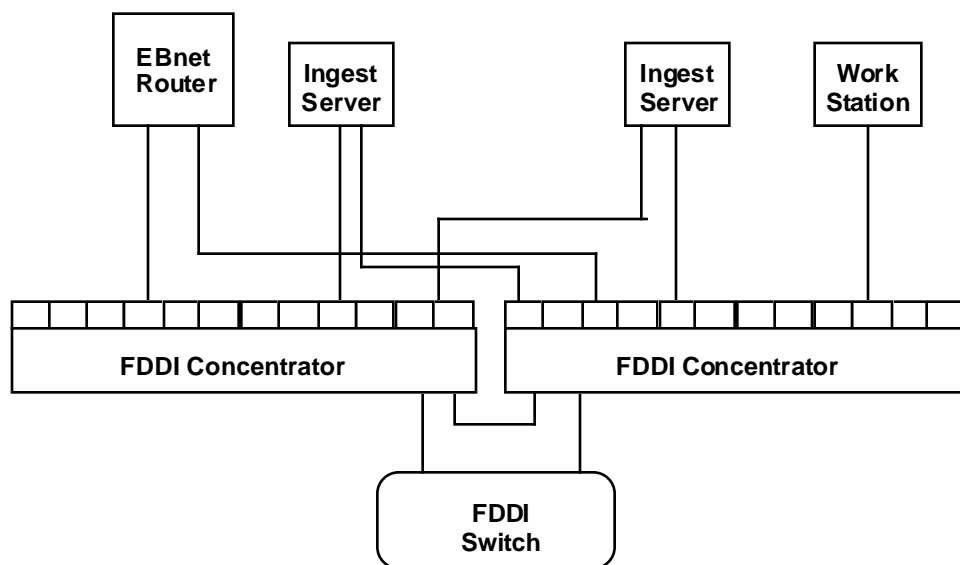
The ASTER GDS will provide data on hard media. The hardware implementation for ASTER media is discussed in the DIPHW CI section of the Data Server volume of the document. Ingest and Data Server share the DIPHW CI, which provides media handling capabilities. Other interfaces (e.g., SDPF and EDOS) provide hard media for contingency transfers and for some

Version 0 migration. The majority of the Release B data transfers will occur through an electronic interface. The mechanism for electronic transfer also varies depending on the data source. Some data sources will notify ECS of the availability of data, which the Ingest Subsystem will get from the source. Others will put data in a predefined location which is then periodically polled by the ingest client software. At Release A, the principal driver in terms of daily electronic L0 data ingest is the data received through the SDPF for the TRMM platform. At Release B, the principal driver of daily electronic L0 data ingest is the data received through EDOS for the AM-1 platform and Landsat-7 LPS L0R data received at EDC. The Moderate Resolution Imaging Spectrometer (MODIS) and Multi-angle Imaging Spectro-Radiometer (MISR) AM-1 instrument data sent to GSFC and LaRC, respectively, comprise the majority of the L0 data to be archived within the Ingest Subsystem for the first several years of ECS operations. The reliable ingest and archiving of this data, and the associated management of Ingest Subsystem resources are the principal drivers in the Ingest Subsystem design. The effects of the interfaces driving the ingest HWCI design on the sizing factors identified in Table 5.1-1 are summarized in Table 5.1-4. Identified impacts on sizing factors and Ingest Subsystem components are estimates of the relative affects that a particular interface to be supported has on the subsystem. Items designated as having a moderate or high impact will drive the capabilities of the individual components as the site supporting that interface.

The Ingest servers and workstation(s) will be directly connected to the DAAC Production FDDI network, as is illustrated in Figure 5.1-2. The Ingest servers will contain dual-attached station (DAS) cards, which will be dual-homed to separate FDDI concentrators. This provides redundancy so that full connectivity will exist to the servers even in the event of a concentrator failure. The workstation(s) will contain single-attached station (SAS) cards and each will be connected to a single concentrator, but they will also be split across concentrators so that they are not all connected to the same unit. The FDDI concentrators are in turn connected to the FDDI switch. (Refer to section 5.2 of Volume 0 for a general description of DAAC networks.)

**Table 5.1-4. Ingest HWCI Interface Drivers**

<b>External I/F</b>	<b>Principal supported functions</b>	<b>Affected sizing factor (high or moderate impact)</b>	<b>Impacted Ingest Subsystem components (high or moderate impact)</b>
EDOS	L0 data ingest	I/O, CPU	Client hosts, working storage, L0 archive repository
SDPF	L0 data ingest	None (low average data volume)	None
NESDIS/DAO (NOAA)	Ancillary data ingest	CPU	Client hosts
Landsat-7	L0R data ingest	I/O, CPU	Client hosts, working storage
SCF	Special Product Ingest	None (low average daily volume)	None
DAACs	V0 data ingest	I/O, CPU	Client host (backupprocessor) and Data Server subsystem hardware
ASTER GDS	LIA/B data ingest	I/O	None (Data Server subsystem hardware)
Users	Derived data set	None (low average daily volume)	None



**Figure 5.1-2. Ingest Network Connectivity**

The Ingest subsystem will have direct interfaces to the L0 data provider (e.g., EBnet at GSFC, LaRC, and EDC). The L0 router(s) will contain a DAS card which will be connected to the two separate FDDI concentrators, which will in turn be connected to the FDDI switch via a physically cabled FDDI ring.

#### 5.1.2.2 Ingest HWCI Component Description

The Ingest HWCI consists of the ingest client hosts, working storage, and L0 archive repository, and is described in the following paragraphs. Estimates for the number of client hosts and other Ingest Subsystem hardware required to support required ingest functions at each DAAC in the Release B time frame are summarized in the DAAC-specific volumes of DID305. Table 5.1-5 summarizes the classes of components which comprise the Ingest HWC.

**Table 5.1-5. Ingest HWCI Component Descriptions**

Component Name	Class/Type	Comments
Client Host	SMP Server W/S / Server W/S	Sites require single or multi-processor mid-level SMP servers
Working Storage	RAID disk	All sites will utilize one or more RAID units. MSFC and JPL will utilize working storage RAID units in the Release-B time frame in place of other data repository hardware
L0 Archive Repository	Archive Robotics	An Automated Tape Library (ATL) or similar robotics unit provides the 1 year storage of L0 data. Required at certain sites in Release B and beyond
	Linear Magnetic Drives	A typical ATL will contain three media drives (e.g., D3 or 3590 tape) per unit. The three media drives perform input, output, and backup functions

## Client Hosts

The client hosts will perform pre-processing of the ingested data sufficient both to ensure the basic quality of the received data and to prepare it for archiving and/or further processing. Checks specific to the transfer mechanism (e.g., list of contents, checksums) will be performed, and successful receipt and storage of the data will be acknowledged to the sender and recorded in the ingest logs. It is assumed that for the Release B configuration most data sets will be in a standard format and that metadata will be read either from the file headers, well defined byte locations within the file, or from separate files provided by the data producers rather than by derivation from the data itself. Metadata to be read and logged or generated upon ingest includes data set name, time of ingest, observation time, and granule id. Required format conversion varies from DAAC to DAAC. Additional CPU capabilities necessary to support these conversions will be determined as the required conversions are identified.

Ingest host I/O capabilities must include support for multiple high-bandwidth internal network and peripheral connections. The required capabilities include technologies such as HiPPI, FDDI, and SCSI II fast/wide for internal network and peripheral connections based on the needs at each site. In particular, some sites implement digitizing/scanning hardware for electronic ingest of hardcopy documents. In addition, sites that require migration of Version 0 data field peripheral devices to support the pre-ECS migration process (using the facilities of the backup host processor).

The types of high performance network connections and data rates to be supported by the client host hardware requires significant CPU capabilities. Candidates for client host hardware are discussed in the DAAC-specific configuration appendix and are based on the ingest I/O rate and volume requirements at each site. In general, the client host CPUs are low to mid-level Symmetric Multi-Processing (SMP) 32-bit machines, capable of supporting multiple network (FDDI) and direct-connect (SCSI II) devices. Commonality of components between subsystems is an SDPS design goal. Therefore, the choice of the platform for the ingest client hosts is one that has been made together with that for the data server specifically, as well as processing and other SDPS subsystems, as appropriate. It is typical that workstations from a given manufacturer can be expanded in their CPU and I/O capabilities within a given platform and without significant changes to the system or application software. CPU and I/O capabilities vary at different sites, but should be supported by the same basic platform which may be upgraded with additional capabilities as increased I/O and processing loads demand.

## Working Storage

Short term working storage provides a staging area for data moving both into the Ingest Subsystem from network connections or ingest peripherals, and out of the Ingest Subsystem to the Processing and Data Server Subsystems. Short term working storage performance and capacity is driven by the requirement that the Ingest Subsystem reliably and efficiently capture large volumes of L0 data as received from EDOS, SDPF, and other sources. The implementation is also driven by the need for high RMA, as L0 data received from these sources must not be lost or delayed in its availability to other ECS subsystems. This function will be implemented using high performance RAID magnetic disks. Arrays will be shared across ingest hosts and will be sized to handle the anticipated ingest volume, the staging of data out of the Ingest Subsystem, and working storage volume necessary for reformatting, metadata validation, and metadata extraction operations.

Other than the higher RMA requirement, the Ingest Subsystem requirements for working storage are largely the same as those for the Data Server Subsystem and can be satisfied with the same type of hardware. Availability will be 0.999 or better for the ingest of L0 data using a strategy encompassing the use of warm spares; the reallocation of working storage space based on data priority within the ingest, processing, and Data Server Subsystems; and the regular and timely transfer of data to long term storage. Working storage devices are RAID 5 units generally with a minimum of 2.0 days worth of space allocated to ingest working storage. These devices will be sized to the data requirements of the individual sites and will support the functions of acquiring, processing, validating, and archiving L0 data. This volume of working storage allows for one day's worth of L0 data to be staged for processing, an additional day's worth available for subsequent ingest, and an additional 25% available to service additional Ingest Subsystem needs (e.g., L0 archive retrieval support, pre-processing, quality checking). Required working storage volume ranges from several hundred megabytes at the smaller sites to several hundred gigabytes at the larger sites.

The working storage at EDC for Landsat-7 LOR data ingest is the exception. Landsat-7 LOR data is not accessed by the ECS Processing subsystem. Furthermore, the equivalent of twelve hours of Landsat-7 LOR data must be ingested and archived by ECS within eight hours. Accordingly, one day's worth of RAID is allocated for initial ingest and associated preprocessing plus 25% contingency.

An estimate of working storage volume required at each DAAC is shown in Table 5.1-6. Additional magnetic disk resources are also supplied within the Ingest Subsystem to support items such as client host operating system and application software and L0 archive database directory information.

**Table 5.1-6. Annual L0 Storage Volumes**

Ingest Sub-system Capabilities by DAAC	Total Daily L0 In-gest Volume (GB/day)	Average Data In-gest Rate (Mbits/sec)*	Short-term (Working) Storage Volume (GB)**	Level 0 Repository (1 year) Volume (GB)***
ASF				
EDC	139.4	12.91	174.25	0****
GSFC	70.4	6.5	176	25,696
JPL	0.27	0.025	0.675	98.6
LaRC	47.6	4.4	119	17,374
NSIDC				
ORNL				

\* Average daily rate = total daily ingest volume over a 24 hour period

\*\* Working storage volume = (daily volume x 2) plus 25%

\*\*\* Repository volume = daily volume x 365

\*\*\*\* The archive repository for Landsat-7 LOR data is supplied by the Data Server Subsystem

## L0 Archive Repository

The Ingest Subsystem, through a combination of short term working storage and long term storage resources, must pass ingested data to other ECS subsystems as required for processing, or other needs of the data system. An additional function of the long term storage portion of the Ingest Subsystem is to store all ingested spacecraft Level 0 data as received from EDOS for a period of one year (plus or minus a delta time, dependent on data type). For Level 0 data without Level 1A counterpart (e.g., MOPITT) the Level 0 data is stored permanently. These functions are accomplished through the use of a data repository which is dedicated to the Ingest Subsystem.

The volumes to be stored at Release B in several of the DAACs are significant enough to require the addition of a robotics-based archive unit as described below. The L0 archive receives data from the short term working storage and stores it in an automated fashion on high density media. The likely access pattern to this data is one of initial sequential ingest, possible reading of some or all of the ingested data within 24 hours or less, and then infrequent subsequent access. This access pattern, along with the data volumes and ingest rates to be supported at some of the Release B sites, lends itself to a helical scan streaming tape technology (e.g., D3) or linear tape (e.g., 3590) in an automated tape library (ATL) configuration. Multiple tape drives would be available for simultaneous read, write, and hot spare capabilities. The configuration of the ingest L0 archive hardware (e.g., drives, media, robotics, interfaces) are the same as that for the Data Server Subsystem repositories to support the goal of commonality in development, operations, and maintenance of ECS hardware and software. The possible exception to this would be the incorporation of additional media drives and interfaces to support the increased RMA requirements of the Ingest L0 archive, but this would not significantly change the implementation of the repository. The repositories at each site will be sized to accommodate sufficient media to store one years worth of L0 data, plus a scaling factor of approximately 5% to account for file storage management overhead and data storage inefficiencies. Table 5.1-6 shows the required annual volume of L0 data to be stored at each site.

### **5.1.3 Failover and Recovery Strategy**

The Ingest Subsystem failure and recovery strategy is implemented to satisfy the RMA requirements associated with science data ingest, which require an operational availability of 0.999 and switchover from a primary to backup capability within 15 minutes. The failover/recovery strategy for the Ingest Subsystem is implemented in several ways:

- Recovery from the failure of individual data transfers is initiated automatically, with a tunable number of automated retries before a failure alert is sent to an operator.
- Recovery from a failure in the primary client host CPU involves detection of the failure, resetting the network address of the backup client host to that of the primary, mounting the dual-ported disks shared by the backup and primary, and other minimal software reconfiguration required to make the backup to look identical to the primary.

There are three types of network failures that may affect the Ingest subsystem:

- If the FDDI cable between a host and the FDDI concentrator is severed or damaged, then a new cable would need to be installed. No other configuration would be required.
- If an individual port on the FDDI concentrator fails, then the attached host must be moved to another port, again with no other configuration required.

- Finally, if the entire concentrator fails, then it will have to be replaced, which can be done rapidly since the units require very little configuration.

Note that the above failures result in service interruption only to the workstation(s). For the Ingest servers, there is no single point of failure for the network for receiving L0 data. Since all Ingest servers are attached to two hubs, they will communicate as normal in the event of a cable or concentrator fault, and the applications will be unaware of and unaffected by the event (e.g., L0 ingest will not be interrupted).

Failures in working storage are accommodated in two ways:

- Redundancy is an inherent design feature of RAID devices, typically allowing the failure of one drive in the unit without any loss of data and minimal performance degradation. Many RAID units also incorporate software that keeps track of soft errors as well as hard failures, allowing drives that may be gradually failing to be replaced prior to a hard failure. Even in the event of a hard failure, the unit can continue to operate while a new drive is installed and rebuilt using the data from the other drives in the unit.
- A catastrophic failure of the RAID unit would require that ingest operations be temporarily limited to high-priority data ingest only. Each Release B site has at least two RAID devices, with one designated current as working storage and one as space available for catchup ingest. Open capacity in the catchup drive could temporarily be used to take over working storage functions.

## Appendix A. Requirements Trace

The TRMM Development (Release A) and AM-1/Landsat-7 Development (Release B) Level 4 requirements listed in the following table reflect the Release B pre-CDR baseline of the RTM database. The text provided below is a subset of that in the RTM data base and is included to aid the reader in mapping requirements to object classes, CSCs, and CIs.

**Table A-1. Requirements Trace (1 of 17)**

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-INS-00010	The INGST CI shall accept Network Ingest Requests to request automated electronic network ingest of a collection of Data. The collection of Data shall describe one or more Data Granules.	InSession
S-INS-00020	The INGST CI shall check the Network Ingest Request to verify that the date/time prior to which the data will remain available is a valid date/time.	InRequest
S-INS-00030	The INGST CI shall authenticate the provider of a Network Ingest Request as an authorized provider of data to be ingested.	CsGateway
S-INS-00040	The INGST CI shall report status to the provider of a Network Ingest Request and to the Error Log indicating successful or unsuccessful authentication of the provider as authorized to submit the request.	CsGateway
S-INS-00050	The INGST CI shall report the following to the MSS event log services: a. Receipt of a network ingest request; b. Response to a network ingest request.	InSession
S-INS-00060	The INGST CI shall report status to the provider of a Network Ingest Request for the following: a. File transfer failure b. File size discrepancies c. Invalid Data Type Identifier d. Missing required metadata e. Metadata parameters out of range f. Data conversion failure g. Failure to archive data h. Inability to transfer data within the specified time window i. Missing required request information j. Successful archive of the data	InSession
S-INS-00062	The INGST CI shall report the following events by means of the CSS Event Logger Service, during the processing of a Network Ingest Request: a. Receipt of an unexpected message from the ingest provider b. Detection of invalid information on a message received from the ingest provider c. Communication failure with the provider of the Ingest Request, as reported to the INGST CI by CSS communication services d. File transfer failures reported to the INGST CI by CSS File Access Service e. Detection of discrepancies between the number of the file(s) received and the specifications in the Ingest Request.	InSession
S-INS-00064	The INGST CI shall report the following events by means of the CSS Event Logger Service, during tests of the network ingest interface between ECS and external data providers: a. Receipt of a message by the Ingest interface b. Start of processing for a valid Ingest Request c. Completion of all processing associated with the Ingest Request	InSession

**Table A-1. Requirements Trace (2 of 17)**

<b>L4 Rqmt ID</b>	<b>L4 Requirement Text</b>	<b>Object Class, CSC, or CI</b>
S-INS-00070	The INGST CI shall provide the capability to periodically check a location accessible to the ESN for the presence of a Delivery Record file describing data to be ingested. The Delivery Record file shall contain the same information as a Network Ingest Request.	InPollingIngestSession
S-INS-00080	The INGST CI shall read a Delivery Record file describing data to be ingested at a location accessible to the ESN and submit a corresponding Network Ingest Request to be processed.	InDAN
S-INS-00083	The INGST CI shall determine the data type for expedited data provided by EDOS.	InPollingIngestSession
S-INS-00085	The INGST CI shall report status to the provider of a polling ingest request (delivery record file) for the following: a. File transfer failure; b. File size discrepancies; c. Invalid data type identifier; d. Missing required metadata; e. Metadata parameters out of range; f. Failure to archive data; g. Missing required request information; h. Successful archive of the data.	InPollingIngestSession
S-INS-00090	The INGST CI shall provide the capability for authorized operations staff to set the period between checking for the presence of Delivery Record files.	InExternalDataProviderInfo
S-INS-00100	The INGST CI shall provide the capability to periodically check a location accessible to the ESN for the presence of data granule files.	InPollingIngestSession
S-INS-00110	The INGST CI shall submit an Polling Ingest Request after detecting the presence of data granule files in a location accessible to the ESN. The request shall contain the file location.	InPollingIngestSession
S-INS-00120	The INGST CI shall provide the capability for authorized operations staff to set the period between checking for the presence of external data granule files.	InExternalDataProviderInfo
S-INS-00130	The INGST CI shall interactively accept Hard Media Ingest Requests from operations staff for data to be ingested from hard media.	InMediaIngest
S-INS-00140	The INGST CI shall check the Hard Media Ingest Request to verify that the Media Type is a type supported by the facility to which the request was submitted.	InMediaIngest
S-INS-00150	The INGST CI shall verify that the External Data Provider specified in a Hard Media Ingest Request is an authorized provider of hard media to be ingested.	InMediaIngest
S-INS-00160	The INGST CI shall authenticate that the Hard Media Ingest Request is input by operations staff authorized to ingest hard media data.	InMediaIngest
S-INS-00165	The INGST CI shall read a Delivery Record file describing data to be ingested to determine the files to be ingested after hard media data transfer.	InDAN

**Table A-1. Requirements Trace (3 of 17)**

<b>L4 Rqmt ID</b>	<b>L4 Requirement Text</b>	<b>Object Class, CSC, or CI</b>
S-INS-00170	The INGST CI shall report Hard Media Ingest Request status to the submitting operations staff for the following: a. Media file transfer failure b. Invalid Data Type Identifier c. Missing required metadata d. Metadata parameters out of range e. Data conversion failure f. Failure to archive data g. Missing file describing media data to be ingested h. Unauthorized hard media provider i. Unauthorized operations staff j. Successful archive of data	InMediaIngest
S-INS-00175	The INGST CI shall report Hard Media Ingest Request status to the MSS event log for the following:  a. Unauthorized hard media provider  b. Unauthorized operations staff	InMediaIngest
S-INS-00180	The INGST CI shall interactively accept Network Ingest Requests from authorized science users for electronic network ingest of a collection of Data from a location accessible via the ESN. The collection of Data shall describe one or more Data Granules.	InInteractiveIngest
S-INS-00187	The INGST CI shall access the Advertising service to determine the availability of a Network Ingest Request service for a given Data Type Identifier.	InDataTypeTemplate
S-INS-00190	The INGST CI shall check the Network Ingest Request to verify that the date/time prior to which the data will remain available is a valid date/time in a Network Ingest Request entered interactively by a science user.	InRequest
S-INS-00200	The INGST CI shall allow a science user to specify the list of granule files in an interactive Network Ingest Request based on a displayed list of existing files stored on magnetic disk.	InInteractiveIngest
S-INS-00205	The INGST CI shall determine the External Data Provider for a Network Ingest Request entered interactively by a science user.	InInteractiveIngest
S-INS-00207	The INGST CI shall automatically determine the data volume for each file in the list of granule files for an interactively entered Network Ingest Request.	InInteractiveIngest
S-INS-00208	The INGST CI shall authenticate that the interactive science user entering a Network Ingest Request is authorized to request ingest of data.	InInteractiveIngest
S-INS-00209	The INGST CI shall report to the Error Log an unauthorized attempt to interactively request ingest of data.	InInteractiveIngest
S-INS-00210	The INGST CI shall allow authorized science users to save the contents of an interactively entered Network Ingest Request in a Delivery Record file with a specified file name.	InInteractiveIngest
S-INS-00220	The INGST CI shall report status to the interactive submitter of a Network Ingest Request for the following: a. File transfer failure b. File size discrepancy c. Invalid Data Type Identifier d. Missing required metadata e. Metadata parameters out of range f. Data conversion failure g. Failure to archive data h. Inability to transfer data within the specified time window i. Unauthorized science user j. Missing required request information k. Successful archive of the data	InInteractiveIngest
S-INS-00221	The INGST CI shall interactively accept Document Ingest Requests from authorized science users for ingest of a single collection of document Data from a location accessible via the ESN. The collection of document Data shall describe one or more document Data Granules.	InInteractiveIngest

**Table A-1. Requirements Trace (4 of 17)**

<b>L4 Rqmt ID</b>	<b>L4 Requirement Text</b>	<b>Object Class, CSC, or CI</b>
S-INS-00222	The INGST CI shall check the Document Ingest Request to verify that the date/time prior to which the data will remain available is a valid date/time in a Document Ingest Request entered interactively by a science user.	InRequest
S-INS-00224	The INGST CI shall allow a science user to specify the list of document granule files in an interactive Document Ingest Request based on a displayed list of existing files stored on magnetic disk.	InInteractiveIngest
S-INS-00225	The INGST CI shall determine the data provider and assign the Priority Information for a Document Ingest Request entered interactively by a science user.	InInteractiveIngest
S-INS-00226	The INGST CI shall automatically determine the data volume for each file in the list of document granule files for an interactively entered Document Ingest Request.	InInteractiveIngest
S-INS-00227	The INGST CI shall authenticate that the interactive science user entering a Document Ingest Request is authorized to request ingest of data.	InInteractiveIngest
S-INS-00228	The INGST CI shall report to the Error Log an unauthorized attempt to interactively request ingest of document data.	InInteractiveIngest
S-INS-00229	The INGST CI shall allow authorized science users to save the contents of an interactively entered Document Ingest Request in a file with a specified file name.	InInteractiveIngest
S-INS-00230	The INGST CI shall report status to the interactive submitter of a Document Ingest Request for the following: a. File transfer failure b. File size discrepancy c. Invalid Data Type Identifier d. Missing required metadata e. Metadata parameters out of range f. Data conversion failure g. Failure to archive data h. Inability to transfer data within the specified time window i. Unauthorized science user j. Missing required request information k. Successful archive of the data	InInteractiveIngest
S-INS-00234	The INGST CI shall access the Advertising service to determine the availability of a Document Ingest Request service for a given Data Type Identifier.	InDataTypeTemplate
S-INS-00235	The INGST CI shall accept ingest Status Requests from science users to determine the status of: a. A specified ongoing Ingest Request, previously submitted by the science user who is requesting status and identified by the ingest Request Identifier b. All of the user's ongoing Ingest Requests	InRequestController
S-INS-00240	The INGST CI shall determine the User Identifier for a science user submitting an ingest Status Request.	InRequestController
S-INS-00250	The INGST CI shall return status on a science user's ongoing Network Ingest Requests, based on User Identifier, to the user.	InRequestController
S-INS-00260	The INGST CI shall provide science users the capability to display the status of the user's ongoing request processing. Displayed status shall include the External Data Provider, ingest Request Identifier, total ingest data volume, and Request State.	InRequestController
S-INS-00270	The INGST CI shall accept ingest Status Requests from authorized operations staff to determine the status of: a. A specified ongoing Ingest Request identified by ingest Request Identifier b. All ongoing Ingest Requests associated with a specified User Identifier c. All ongoing Ingest Requests	InRequestController

**Table A-1. Requirements Trace (5 of 17)**

<b>L4 Rqmt ID</b>	<b>L4 Requirement Text</b>	<b>Object Class, CSC, or CI</b>
S-INS-00280	The INGST CI shall determine the User Identifier for an operations staff member submitting an ingest Status Request.	InRequestController
S-INS-00290	The INGST CI shall authenticate the User Identifier of operations staff requesting status on all ongoing Ingest Requests.	InRequestController
S-INS-00295	The INGST CI shall return an error status to the requester and log information in the Error Log if status is requested on ongoing Ingest Requests from an unauthorized requester.	InRequestController
S-INS-00300	The INGST CI shall return status on ongoing Ingest Requests to an authorized operations staff member.	InRequestController
S-INS-00310	The INGST CI shall provide authorized operations staff the capability to view the status of ongoing ingest processing. Displayed status shall include the External Data Provider, ingest Request Identifier, total ingest data volume, and Request State.	InRequestController
S-INS-00315	The INGST CI shall provide the capability for authorized operations staff to select status of ongoing Ingest Request processing for viewing by means of the External Data Provider.	InRequestController
S-INS-00316	The INGST CI shall accept an Ingest Request from authorized applications.	InRequestManager
S-INS-00317	The INGST CI shall authenticate the User Identifier of an application submitting an Ingest Request.	InRequestManager
S-INS-00318	The INGST CI shall determine the Priority Information for each Ingest Request based on the External Data Provider and the requested ingest priority for the request.	InRequestManager
S-INS-00319	The INGST CI shall add a submitted Ingest Request to a list of Ingest Requests sorted by Priority Information.	InRequestManager
S-INS-00320	The INGST CI shall select an Ingest Request for processing based on the priorities of current requests so long as the number of requests concurrently processed is less than a threshold specified by operations staff. Requests of equal priority will be processed first-in, first-out.	InRequestManager
S-INS-00321	The INGST CI shall advertise available Interactive Network Ingest services.	InInteractiveIngest
S-INS-00325	The INGST CI shall determine the ingest start/stop dates and times for all ingested data.	InRequest
S-INS-00330	The INGST CI shall determine the Data Type Identifier for a set of ingested files, whenever the identifier was not provided in the Ingest Request.	Deleted by pending CCR
S-INS-00340	The INGST CI shall report status on processing of an Ingest Request to the Error Log for the following: a. File transfer failure b. File size discrepancy c. Invalid Data Type Identifier d. Missing required metadata e. Metadata parameters out of range f. Metadata extraction failure g. Data conversion failure h. Data reformatting failure i. Failure to archive data j. Inability to transfer data within the specified time window k. Missing required request information l. Unauthorized Ingest Request submitter m. Successful archive of the data	InRequest

**Table A-1. Requirements Trace (6 of 17)**

<b>L4 Rqmt ID</b>	<b>L4 Requirement Text</b>	<b>Object Class, CSC, or CI</b>
S-INS-00345	The INGST CI shall report status on the performance of ingest requests to the MSS with the following information: a. file transfer duration b. file processing duration c. data insert duration	InGranuleAsynch_S
S-INS-00350	The INGST CI shall accept an ingest Cancellation Request from authorized operations staff to cancel an ongoing ingest request, specifying the ingest Request Identifier.	InRequestController
S-INS-00355	The INGST CI shall accept an ingest Suspension Request from authorized operations staff to suspend ongoing ingest request processing for a specified ingest Request Identifier, to suspend all ongoing ingest request processing from a specified External Data Provider, or to suspend all ongoing ingest request processing.	InRequestController
S-INS-00357	The INGST CI shall accept an ingest Resumption Request from authorized operations staff to resume ongoing ingest request processing for a specified ingest Request Identifier, to resume all ongoing ingest request processing from a specified External Data Provider, or to resume all ongoing ingest request processing.	InRequestController
S-INS-00360	The INGST CI shall authenticate the User Identifier of operations staff submitting an ingest Cancellation Request.	InRequestController
S-INS-00363	The INGST CI shall authenticate the User Identifier of operations staff submitting an ingest Suspension Request or ingest Resumption Request.	InRequestController
S-INS-00364	The INGST CI shall accept an ingest Cancellation Request from authorized applications to cancel an ongoing Ingest Request, specifying the Request Identifier.	InRequestManager
S-INS-00365	The INGST CI shall accept an ingest Suspension Request from authorized applications to suspend ongoing ingest request processing for a specified Request Identifier, to suspend all ongoing ingest request processing from a specified External Data Provider, or to suspend all ongoing ingest request processing.	InRequestManager
S-INS-00367	The INGST CI shall accept an ingest Resumption Request from authorized applications to resume ongoing ingest request processing for a specified Request Identifier, to resume all ongoing ingest request processing from a specified External Data Provider, or to resume all ongoing ingest request processing.	InRequestManager
S-INS-00369	The INGST CI shall authenticate the User Identifier of an application submitting an ingest Cancellation Request.	InRequestManager
S-INS-00370	The INGST CI shall authenticate the User Identifier of an application submitting an ingest Suspension Request or ingest Resumption Request.	InRequestManager
S-INS-00380	The INGST CI shall provide authorized operations staff the capability to set thresholds for: a. Total number of Ingest Requests to process concurrently b. Number of Ingest Requests for each External Data Provider to process concurrently c. Total volume of data to ingest concurrently d. Volume of data for each External Data Provider to ingest concurrently e. Number of data transfer retry attempts for each external interface to ECS	InExternalDataProvid erInfo
S-INS-00390	The INGST CI shall authenticate the User Identifier of operations staff requesting to set thresholds for concurrent ingest processing.	InExternalDataProvid erInfo

**Table A-1. Requirements Trace (7 of 17)**

<b>L4 Rqmt ID</b>	<b>L4 Requirement Text</b>	<b>Object Class, CSC, or CI</b>
S-INS-00392	The INGST CI shall report status on ingest Cancellation Requests to the requesting operations staff and to the Error Log for the following: a. Unauthorized requester b. Invalid ingest Request Identifier c. Unable to cancel specified Ingest Request	InRequestController
S-INS-00393	The INGST CI shall report status on ingest Suspension Requests to the requesting operations staff and to the Error Log for the following:  a._Unauthorized requester  b._Invalid ingest Request Identifier  c._Unable to suspend specified Ingest Request(s)	InRequestController
S-INS-00394	The INGST CI shall report status on ingest Resumption Requests to the requesting operations staff and to the Error Log for the following: a. Unauthorized requester b. Invalid ingest Request Identifier	InRequestController
S-INS-00395	The INGST CI shall report status on ingest threshold setup Requests to the requesting operations staff and to the Error Log for the following: a. Unauthorized requester b. Invalid ingest Request Identifier c. Unable to suspend specified Ingest Request(s)	InRequestController
S-INS-00396	The INGST CI shall report status on ingest Cancellation Requests to the requesting application and to the Error Log for the following: a. Unauthorized requester b. Invalid ingest Request Identifier c. Unable to suspend specified Ingest Request(s)	InRequestManager
S-INS-00397	The INGST CI shall report status on ingest Suspension Requests to the requesting application and to the Error Log for the following:  a._Unauthorized requester  b._Invalid ingest Request Identifier  c._Unable to suspend specified Ingest Request(s)	InRequestManager
S-INS-00398	The INGST CI shall report status on ingest Resumption Requests to the requesting application and to the Error Log for the following:  a._Unauthorized requester  b._Invalid ingest Request Identifier	InRequestManager
S-INS-00400	The INGST CI shall convert ingested data into a form accepted by the SDSRV CI / DDSRV CI, for following data types: a. NMC GRIB data.	InGRIBData
S-INS-00401	The INGST CI shall convert ingested data into a form accepted by the SDSRV CI/DDSRV CI.	InDataPreprocessTask
S-INS-00402	The INGST CI shall reformat ingested data into a form accepted by the SDSRV CI/DDSRV CI, as needed.	InDataPreprocessTask
S-INS-00403	The INGST CI shall perform the following metadata conversions: a. PB5 time into ECS standard date / time format; b. Binary integer values into ASCII integer format; c. Binary floating point values into ASCII floating point format.	InMetadata
S-INS-00404	The INGST CI shall extract metadata from ingested data into a form accepted by the Science Data Server / Document Data Server, as needed, for the following categories of data: a. Metadata parameters stored by parameter byte order and parameter byte length; b. Metadata parameters stored in PVL format; c. Metadata parameters stored in HDF format; d. Dataset-specific metadata formats	InMetadata
S-INS-00405	The INGST CI shall append the following ingest-specific metadata to metadata corresponding to ingested data: a. Ingest start date and time b. Ingest stop date and time c. Metadata parameter check status d. Total data volume	InMetadata

**Table A-1. Requirements Trace (8 of 17)**

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-INS-00406	The INGST CI shall check selected parameters from extracted metadata to verify:  a. Metadata parameters stored in a dataset specific format b. For numeric metadata parameters limited by a range of values, that parameter values lie within the specified range c. For metadata parameters with values limited to a set of discrete values, that parameter values are listed in the specified set d. That the metadata parameter syntax is correct e. For metadata containing parameters describing the data size, that the data size is correct (within a specified tolerance) f. That date / time values include a valid month, day of month, hour, minute, and second g. That date / time values include a year value within a range specific for that date / time value	InMetadata
S-INS-00408	For each data granule specified in an Ingest Request the INGST CI shall determine by means of an Advertisement the appropriate SDSRV CI/DDSRV CI in which to store the data granule.	InDataServerInsertionTask
S-INS-00409	The INGST CI shall provide the capability to request storage of a data granule by means of a Data Insert Request to the SDSRV CI/ DDSRV CI associated with the type of the data granule.	InDataServerInsertionTask
S-INS-00410	The INGST CI shall provide the capability to electronically transfer data to be ingested via the ESN into a specified ECS storage location.	InDataTransferTask
S-INS-00420	The INGST CI shall provide the capability for an external application to transfer data to be ingested into a specified ECS storage location.	InInteractiveIngest
S-INS-00425	The INGST CI shall provide the capability to request transfer of data from an 8mm tape.	InMediaIngest
S-INS-00430	The INGST CI shall provide the capability by means of a Working Storage Allocation Request to the Data Server to allocate storage space for data to be transferred to satisfy an ingest request.	InDataTransferTask
S-INS-00440	The INGST CI shall estimate whether data may complete transfer before the date/time prior to which the data will remain available.	InRequest
S-INS-00450	The INGST CI shall retry transfer of data from the External Data Provider N times before the ingest request is failed, where N is a number specified by operations staff.	InDataTransferTask
S-INS-00455	Operations staff shall contact the network operations staff and External Data Provider operations staff to resolve data transfer problems that are not handled automatically.	N/A (Operational)
S-INS-00460	The INGST CI shall determine the size of each file transferred to ECS whenever file sizes are specified in the corresponding Ingest Request.	InFile
S-INS-00470	The INGST CI shall compare the size of each file after data transfer to ECS with file sizes specified in the corresponding Ingest Request.	InFile
S-INS-00480	The INGST CI shall verify that all files specified in an Ingest Request are successfully transferred to ECS.	InDataTransferTask

**Table A-1. Requirements Trace (9 of 17)**

<b>L4 Rqmt ID</b>	<b>L4 Requirement Text</b>	<b>Object Class, CSC, or CI</b>
S-INS-00490	The INGST CI shall log the following information in an Ingest History Log for each received Ingest Request: a. Ingest start/stop dates and times b. Ingest Request Identifier c. External Data Provider d. Final Service Request Status e. Data Type Identifiers f. Ingest data volume g. # of data sets h. # of data files	InRequest
S-INS-00500	The INGST CI shall provide operations staff the capability to view selected entries from the Ingest History Log.	InHistoryLog
S-INS-00510	The INGST CI shall provide the capability to select Ingest History Log entries for viewing by the following parameters:  a. Ingest start/stop dates and times b. External Data Provider c. Data Type Identifier d. Final Service Request Status e. Test or operational mode	InHistoryLog
S-INS-00520	The INGST CI shall ingest data, provided by the SDPF, from the ESN into the LaRC DAAC, using a file transfer protocol.	InSession
S-INS-00530	The INGST CI shall ingest data, provided by the SDPF, from physical media into the LaRC DAAC as a backup transfer mechanism.	InMediaIngest
S-INS-00540	The INGST CI shall ingest data, provided by the SDPF, from the ESN into the MSFC DAAC using a file transfer protocol.	InSession
S-INS-00550	The INGST CI shall ingest data, provided by the SDPF, from physical media into the MSFC DAAC as a backup transfer mechanism.	InMediaIngest
S-INS-00560	The INGST CI shall ingest Data, provided by the TSDIS, from the ESN into the GSFC DAAC using a file transfer protocol.	InSession
S-INS-00570	The INGST CI shall ingest Data, provided by the TSDIS, from the ESN into the MSFC DAAC using a file transfer protocol.	InSession
S-INS-00580	The INGST CI shall ingest Data, provided by the EDOS, from the ESN into the GSFC DAAC using a file transfer protocol.	InPollingIngestSession
S-INS-00590	The INGST CI shall ingest Data, provided by the EDOS, from the ESN into the LaRC DAAC using a file transfer protocol.	InPollingIngestSession
S-INS-00600	The INGST CI shall ingest Data, provided by the EDOS, from physical media at the GSFC DAAC as a backup transfer mechanism.	InMediaIngest
S-INS-00610	The INGST CI shall ingest Data, provided by the EDOS, from physical media at the LaRC DAAC as a backup transfer mechanism.	InMediaIngest
S-INS-00620	The INGST CI shall ingest data, provided by the DAO, from the ESN into the LaRC DAAC using a file transfer protocol.	InPollingIngestSession
S-INS-00630	The INGST CI shall ingest data, provided by NESDIS, from the ESN into the LaRC DAAC using a file transfer protocol.	InPollingIngestSession
S-INS-00640	The INGST CI shall ingest data, provided by the DAO, from the ESN into the GSFC DAAC using a file transfer protocol.	InPollingIngestSession
S-INS-00645	The INGST CI shall ingest Data, provided by the NMC, from the LAN into the GSFC DAAC using a file transfer protocol.	InPollingIngestSession
S-INS-00650	The INGST CI shall ingest data, provided by the DAO, from the ESN into the EDC DAAC using a file transfer protocol.	InPollingIngestSession

**Table A-1. Requirements Trace (10 of 17)**

<b>L4 Rqmt ID</b>	<b>L4 Requirement Text</b>	<b>Object Class, CSC, or CI</b>
S-INS-00670	The INGST CI shall ingest Data, provided by an SCF, from the ESN into the MSFC DAAC using a file transfer protocol.	InInteractiveIngest
S-INS-00680	The INGST CI shall ingest Data, provided by an SCF, from the ESN into the LaRC DAAC using a file transfer protocol.	InInteractiveIngest
S-INS-00682	The INGST CI shall ingest Data, provided by an SCF, from the LAN into the GSFC DAAC using a file transfer protocol.	InInteractiveIngest
S-INS-00684	The INGST CI shall ingest Data, provided by an SCF, from the LAN into the JPL DAAC using a file transfer protocol.	InInteractiveIngest
S-INS-00720	The INGST CI shall ingest data, provided by the EOC, from the ESN using a file transfer protocol.	InSession
S-INS-00730	The INGST CI shall ingest data, provided by the FDF, from the ESN into the GSFC DAAC using a file transfer protocol.	InSession
S-INS-00740	The INGST CI shall accept a TBD request for Repaired Orbit Data.	Deleted by CCR 96-0104
S-INS-00780	The INGST CI shall ingest data, provided by the Landsat 7 Processing Facility (LPS), from the ESN into the EDC DAAC using a file transfer protocol.	InSession
S-INS-00785	The INGST CI shall ingest Data, provided by the Landsat 7 Image Assessment System (IAS), from the LAN into the EDC DAAC using a file transfer protocol.	InInteractiveIngest
S-INS-00787	The INGST CI shall ingest Data, provided by the Landsat 7 International Ground Stations (IGSs), into the EDC DAAC on 8 mm cartridge tape.	InMediaIngest
S-INS-00790	The INGST CI shall ingest data, received on physical media from the ASTER GDS, into the EDC DAAC.	InMediaIngest
S-INS-00800	The INGST CI shall ingest Data, provided by Version 0, from the LaRC DAAC using a file transfer protocol.	InPollingIngestSession
S-INS-00810	The INGST CI shall ingest Data, provided by Version 0, from the GSFC DAAC on 8mm tape.	InMediaIngest
S-INS-00830	The INGST CI shall ingest Data, provided by Version 0, from the MSFC DAAC on 8mm tape.	InMediaIngest
S-INS-00840	The INGST CI shall ingest data provided by ADEOS II/SeaWinds into the JPL DAAC.	InPollingIngestSession
S-INS-00841	The INGST CI shall ingest data, provided by RADARSAT Geophysical Processing System (RGPS), into the ASF DAAC via file transfer protocol.	InSession
S-INS-00843	The INGST CI shall ingest data, provided by the Acquisition Planning System (APS), into the ASF DAAC via file transfer protocol.	InSession
S-INS-00845	The INGST CI shall ingest data, provided by the Product Verification System (PVS), into the ASF DAAC via file transfer protocol.	InSession
S-INS-00847	The INGST CI shall ingest data, provided by the Production Planning System (PPS), into the ASF DAAC via file transfer protocol.	InSession

**Table A-1. Requirements Trace (11 of 17)**

<b>L4 Rqmt ID</b>	<b>L4 Requirement Text</b>	<b>Object Class, CSC, or CI</b>
S-INS-00849	The INGST CI shall ingest data, provided by the Flight Agency Interface (FAIF), into the ASF DAAC via file transfer protocol.	InSession
S-INS-00850	The INGST CI shall ingest Data, provided by SAGE III, into the LaRC DAAC.	InPollingIngestSession
S-INS-00852	The INGST CI shall ingest Data, provided by ACRIM, into the LaRC DAAC.	InPollingIngestSession
S-INS-00854	The INGST CI shall ingest Data, provided by the ASF Receiving Ground Station (RGS) via a network interface using a file transfer protocol.	InSession
S-INS-00856	The INGST CI shall ingest Data, provided by the ASF SAR Processing System (SPS) via a network interface using a file transfer protocol.	InSession
S-INS-00870	The ICLHW CI at the GSFC DAAC shall be capable of ingesting data .for EDOS/ECOM interface testing.	ICLHW
S-INS-00880	The ICLHW CI at the LaRC DAAC shall be capable of ingesting data for EDOS/ECOM interface testing.	ICLHW
S-INS-00900	The INGST CI at the GSFC DAAC shall be capable of 200 percent expansion in throughput without architecture or design change.	InRequestManager
S-INS-00910	The INGST CI at the LaRC DAAC shall be capable of 200 percent expansion in throughput without architecture or design change.	InRequestManager
S-INS-00920	The INGST CI at the MSFC DAAC shall be capable of 200 percent expansion in throughput without architecture or design change.	InRequestManager
S-INS-00925	The INGST CI at the EDC DAAC shall be capable of 200 percent expansion in throughput without architecture or design change.	InRequestManager
S-INS-00927	The INGST CI at the NSIDC DAAC shall be capable of 200 percent expansion in throughput without architecture or design change.	InRequestManager
S-INS-00929	The INGST CI at the ASF DAAC shall be capable of 200 percent expansion in throughput without architecture or design change.	InRequestManager
S-INS-00930	The INGST CI at the JPL DAAC shall be capable of 200 percent expansion in throughput without architecture or design change.	InRequestManager
S-INS-00990	The ICLHW CI at the LaRC DAAC shall be capable of ingesting data from the SDPF at the nominal daily rate specified in Table E-3 of Appendix E of the current version of 304-CD-002 for Release A.	ICLHW
S-INS-01000	The ICLHW CI at the LaRC DAAC shall be capable of ingesting data from the SDPF at a maximum daily rate that is three times the nominal rate specified in Table E-3 of Appendix E of the current version of 304-CD-002 for Release A.	ICLHW
S-INS-01030	The ICLHW CI at the LaRC DAAC shall be capable of ingesting data, by network data transfer from the NESDIS, at the nominal daily rate specified in Table E-3 of Appendix E of the current version of 304-CD-002 for Release A.	ICLHW
S-INS-01035	The ICLHW CI at the LaRC DAAC shall be capable of ingesting data, by network data transfer from NESDIS, at the nominal daily rate specified in Tables E-3a and E-3b of appendix E of the current version of 304-CD-005 for Release B.	ICLHW

**Table A-1. Requirements Trace (12 of 17)**

<b>L4 Rqmt ID</b>	<b>L4 Requirement Text</b>	<b>Object Class, CSC, or CI</b>
S-INS-01040	The INGST CI at the LaRC DAAC shall be capable of receiving data from the SDPF once per day within 24 hours of the last acquisition Client Session.	ICLHW
S-INS-01050	The ICLHW CI at the MSFC DAAC shall be capable of ingesting data from the SDPF at the nominal daily rate specified in Table E-3 of Appendix E of the current version of 304-CD-002 for Release A.	ICLHW
S-INS-01060	The ICLHW CI at the MSFC DAAC shall be capable of ingesting data from the SDPF at a maximum daily rate that is three times the nominal rate specified in Table E-3 of Appendix E of the current version of 304-CD-002 for Release A.	ICLHW
S-INS-01100	The INGST CI at the MSFC DAAC shall be capable of receiving data set from the SDPF once per day within 24 hours of the last acquisition Client Session.	ICLHW
S-INS-01136	The ICLHW CI at the GSFC DAAC shall be capable of ingesting data from the DAO at the nominal daily rate specified in Table E-3 of Appendix E of the current version of 304-CD-002 for Release A.	ICLHW
S-INS-01137	The ICLHW CI at the GSFC DAAC shall be capable of ingesting data from the NMC at the nominal daily rate specified in Tables E-3a and E-3b of Appendix E of the current version of 304-CD-005 for Release B.	ICLHW
S-INS-01138	The ICLHW CI at the LaRC DAAC shall be capable of ingesting data from the DAO at the nominal daily rate specified in Table E-3 of Appendix E of the current version of 304-CD-002 for Release A.	ICLHW
S-INS-01140	The ICLHW CI at the LaRC DAAC shall be capable of ingesting data from the NMC at the nominal daily rate specified in Tables E-3a and E-3b of Appendix E of the current version of 304-CD-005 for Release B.	ICLHW
S-INS-02000	The INGST CI shall interactively accept Document Scanning/Digitizing Requests from authorized operations staff for hard copy media to be ingested.	InInteractiveIngest
S-INS-02010	The INGST CI shall authenticate that the Document Scanning/Digitizing Request is input by operations staff authorized to ingest hard copy media.	InInteractiveIngest
S-INS-02020	The INGST CI shall verify that the External Data Provider specified in a Document Scanning/Digitizing Request is an authorized provider of hard copy media to be ingested.	InInteractiveIngest
S-INS-02030	The INGST CI shall automatically determine the data volume for each scanned or digitized file resulting from an interactively entered Document Scanning/Digitizing Request.	InInteractiveIngest
S-INS-02040	The INGST CI shall report to the Error Log an unauthorized attempt to interactively request ingest of hard copy media.	InInteractiveIngest
S-INS-02050	The INGST CI shall report Document Scanning/Digitizing Request status to the submitting operations staff for the following:  a._Hard copy scanning/digitizing failure  b._Invalid Data Type Identifier  c._Missing required metadata  d._Metadata parameters out of range  e._Failure to archive data  f._Unauthorized hard copy media provider  g._Unauthorized operations staff  h._Successful archive of data	InInteractiveIngest

**Table A-1. Requirements Trace (13 of 17)**

<b>L4 Rqmt ID</b>	<b>L4 Requirement Text</b>	<b>Object Class, CSC, or CI</b>
S-INS-03103	"The INGST CI shall extract metadata from ingested data into a form accepted by the Science Data Server/Document Data Server, as needed, for the following categories of data:" a. Metadata parameters stored in a data-set-specific format	InDataPreprocessTask
S-INS-03200	The INGST CI shall be capable of operating in an off-line (test) mode.	InRequestManager
S-INS-03200	The INGST CI shall be capable of operating in an off-line (test) mode.	InServer
S-INS-03210	The INGST CI shall be capable of accessing test data sets when operating in off-line (test) mode.	InRequestManager
S-INS-60110	The ICLHW CI shall support the hardware resource requirements of the INGST CI and its interface requirements with the operations staff.	ICLHW
S-INS-60150	The ICLHW CI shall have provision for Initialization, Recovery, and an orderly shutdown.	ICLHW
S-INS-60160	Startup and initialization of the ICLHW CI shall be completed within 30 minutes (TBR).	ICLHW
S-INS-60170	Shutdown of the ICLHW CI shall be completed within 30 minutes (TBR).	ICLHW
S-INS-60190	The ICLHW CI shall have a status monitoring capability.	ICLHW
S-INS-60210	The INGST CI shall support a maximum of 300 transactions per day, as specified for each release and corresponding DAAC sites in Table E-3e of the current version of 304-CD-002 for Release A.	ICLHW
S-INS-60310	The ICLHW CI shall be capable of operating in a 24 hour per day, 7 days a week mode.	ICLHW
S-INS-60320	The ICLHW CI shall be configured to support the SDPS function of Receiving Science Data's Availability requirement of .999 and Mean Down Time requirement of < 2 hours during times of staffed operation.	ICLHW
S-INS-60325	The ICLHW CI shall be configured to support the SDPS function of Metadata Ingest and Update's Availability requirement of .96 and Mean Down Time requirement of 4 hours or less.	ICLHW
S-INS-60326	The maximum down time of the ICLHW CI shall not exceed twice the required MDT in 99 percent of failure occurrences.	ICLHW
S-INS-60330	The ICLHW CI elements and components shall include the on-line (operational mode) and off-line (test mode) fault detection and isolation capabilities required to achieve the specified operational availability requirements.	ICLHW
S-INS-60410	The ICLHW CI shall provide maintenance interfaces to support the function of System Maintenance.	ICLHW
S-INS-60420	The ICLHW CI shall provide operations interfaces to support the function of System Maintenance.	ICLHW
S-INS-60430	The ICLHW CI platforms shall have provision for interfacing with one or more Local Area Networks (LANs).	ICLHW
S-INS-60510	The electrical power requirements for ICLHW CI equipment shall be in accordance with and the ECS Facilities Plan (DID 302/DV2).	ICLHW

**Table A-1. Requirements Trace (14 of 17)**

<b>L4 Rqmt ID</b>	<b>L4 Requirement Text</b>	<b>Object Class, CSC, or CI</b>
S-INS-60540	The air conditioning requirements for ICLHW CI equipment shall be in accordance with the ECS Facilities Plan (DID 302/DV2).	ICLHW
S-INS-60550	The grounding requirements for ICLHW CI equipment shall be in accordance with ECS Facilities Plan (DID 302/DV2).	ICLHW
S-INS-60560	The fire alarm requirements for ICLHW CI equipment shall be in accordance with ECS Facilities Plan (DID 302/DV2).	ICLHW
S-INS-60570	The acoustical requirements for ICLHW CI equipment shall be in accordance with ECS Facilities Plan (DID 302/DV2).	ICLHW
S-INS-60580	The physical interface requirements between ICLHW CI equipment and the facility shall be in accordance with ECS Facilities Plan (DID 302/DV2).	ICLHW
S-INS-60590	The footprint size and the physical layout of ICLHW CI equipment shall be in accordance with the ECS Facilities Plan (DID 302/DV2).	ICLHW
S-INS-60605	The ICLHW CI shall support test activities throughout the development phase.	ICLHW
S-INS-60610	The following testing shall be performed on the ICLHW CI: a. Unit Testing b. Subsystem testing c. Integration & Testing d. End-to-End testing	ICLHW
S-INS-60620	Internal testing shall be performed on the ICLHW CI which includes tests of hardware functions, and integration testing with other SDPS subsystems.	ICLHW
S-INS-60630	Internal testing shall be performed on the ICLHW CI to verify the internal interfaces to the Data Management, Client, Data Server, Planning, and Data Processing subsystems.	ICLHW
S-INS-60640	Each ICLHW CI element shall be capable of supporting end-to-end test and verification activities of the EOS program including during the pre-launch, spacecraft verification, and instrument verification phases.	ICLHW
S-INS-60650	The ICLHW CI shall be capable of being monitored during testing.	ICLHW
S-INS-60660	The ICLHW CI shall include the on-line (operational mode) and off-line (test mode) fault detection and isolation capabilities required to achieve the specified operational availability requirements.	ICLHW
S-INS-60733	The ICLHW CI shall contain the storage and interface resources to support the ingest functions for the Landsat 7 Processing System interface at EDC.	ICLHW
S-INS-60736	The ICLHW CI at the GSFC DAAC shall be sized to store and maintain the volume of EDOS data for a 1 year period of time as specified in Appendix E (Section E.1 Table E-1, Section E.2 Table E-2, and Section E.3 Tables E-3a and E-3b) of the current version of 304-CD-005.	ICLHW
S-INS-60740	The ICLHW CI at the LaRC DAAC shall be sized to store and maintain the volume of SDPF data for a 1 year period of time as specified in Table E-3 of Appendix E of the current version of 304-CD-002 for Release A.	ICLHW

**Table A-1. Requirements Trace (15 of 17)**

<b>L4 Rqmt ID</b>	<b>L4 Requirement Text</b>	<b>Object Class, CSC, or CI</b>
S-INS-60741	The ICLHW CI at the LaRC DAAC shall be sized to store and maintain the volume of EDOS data for a 1-year period of time as specified in Appendix E (Section E.1, Table E-1, Section E.2 Table E-2, and Section E.3 Tables E-3a and E-3b) of the current version of 304-CD-005.	ICLHW
S-INS-60745	The ICLHW CI at the MSFC DAAC shall be sized to store and maintain the volume of SDPF data for a 1 year period of time as specified in Table E-3 of Appendix E of the current version of 304-CD-002 for Release A.	ICLHW
S-INS-60746	The ICLHW CI at the JPL DAAC shall be sized to store and maintain the volume of ADEOS II data for a 1-year period of time as specified in Appendix E (Section E.1 Table E-1, Section E.2 Table E-2, and Section E.3 Tables E-3a and E-3b) of the current version of 304-CD-005.	ICLHW
S-INS-60748	The ICLHW CI at the JPL DAAC shall be sized to store and maintain the volume of ALT-RADAR data for a 1-year period of time as specified in Appendix E (Section E.1 Table E-1, Section E.2 Table E.2 Table E-2, and Section E.3 Tables E-3a and E-3b) of the current version of 304-CD-005.	ICLHW
S-INS-60750	The ICLHW CI at the GSFC DAAC shall be sized to temporarily store ingest data to support early testing of the EDOS interface.	ICLHW
S-INS-60751	The ICLHW CI at the GSFC DAAC shall be sized to temporarily store the volume of EDOS data as specified in Appendix E (Section E.1 Table E-1, Section E.2 Table E-2, and Section E.3 Tables E-3a and E-3b) of the current version of 304-CD-005.	ICLHW
S-INS-60755	The ICLHW CI at the LaRC DAAC shall be sized to temporarily store two times the daily volume of SDPF data as specified in Table E-3 of Appendix E of the current version of 304-CD-002 for Release A.	ICLHW
S-INS-60756	The ICLHW CI at the LaRC DAAC shall be sized to temporarily store the volume of EDOS data as specified in Appendix E (Section E.1 Table E-1, Section E.2 Table E-2, and Section E.3 Tables E-3a and E-3b) of the current version of 304-CD-005.	ICLHW
S-INS-60760	The ICLHW CI at the MSFC DAAC shall be sized to temporarily store two times the daily volume of SDPF data as specified in Table E-3 of Appendix E of the current version of 304-CD-002 for Release A.	ICLHW
S-INS-60765	The ICLHW CI shall have a switchover time from the primary science data receipt capability to a backup capability of 15 minutes or less.	ICLHW
S-INS-60770	The ICLHW CI at the EDC DAAC shall be sized to temporarily store the volume of Landsat 7 data as specified in Appendix E (Section E.1 Table E-1, Section E.2 Table E-2, and Section E.3 Tables E-3a and E-3b) of the current version of 304-CD-005.	ICLHW
S-INS-60771	The ICLHW CI at the JPL DAAC shall be sized to temporarily store the volume of ALT-RADAR data as specified in Appendix E (Section E.1 Table E-1, Section E.2 Table E-2, and Section E.3 Tables E-3a and E-3b) of the current version of 304-CD-005.	ICLHW

**Table A-1. Requirements Trace (16 of 17)**

<b>L4 Rqmt ID</b>	<b>L4 Requirement Text</b>	<b>Object Class, CSC, or CI</b>
S-INS-60772	The ICLHW CI at the JPL DAAC shall be sized to temporarily store the volume of ADEOS II data as specified in Appendix E (Section E.1, Table E-1, Section E.2 Table E-2, and Section E.3 Tables E-3a and E-3b) of the current version of 304-CD-005.	ICLHW
S-INS-60810	The operating system for each UNIX platform in the ICLHW CI shall conform to the POSIX.2 standard.	ICLHW
S-INS-60820	The ICLHW CI POSIX.2 compliant platform shall have the following utilities installed at a minimum: perl, emacs, gzip, tar, imake, prof, gprof, nm.	ICLHW
S-INS-60830	The ICLHW CI POSIX.2 compliant platform shall have the following POSIX.2 user Portability Utilities installed at a minimum: man, vi.	ICLHW
S-INS-60840	The ICLHW CI POSIX.2 compliant platform shall have the following POSIX.2 Software Development Utilities installed at a minimum: make.	ICLHW
S-INS-60850	The ICLHW CI POSIX.2 compliant platform shall have the following POSIX.2 C-Language Development Utilities installed at a minimum: lex, yacc.	ICLHW
S-INS-60860	The ICLHW CI POSIX.2 compliant platform shall have the following Unix shells installed at a minimum: C shell, Bourne shell, Korn shell.	ICLHW
S-INS-60870	The ICLHW CI POSIX.2 compliant platform shall have on-line documentation or printed documentation for each installed tool.	ICLHW
S-INS-60880	The ICLHW CI POSIX.2 compliant platform shall have installed one or more development environment supporting the following languages:a. Cb. C++	ICLHW
S-INS-60890	Each development environment associated with the POSIX.2 compliant platform in the ICLHW CI shall have the capability to compile and link strictly conformant POSIX-compliant source code.	ICLHW
S-INS-60895	Each development environment associated with the POSIX.2 compliant platform in the ICLHWCI shall have an interactive source level debugger for ECS supported languages.	ICLHW
S-INS-60900	The INGST CI shall provide the necessary hardware/software to perform scanning and/or digitizing of hardcopy documents for the purpose of inputting document request from authorized users.	ICLHW
S-INS-61000	The ICLHW CI at the GSFC DAAC shall be capable of ingesting data from the EDOS at the nominal daily rate specified in Appendix E (Section E.1, Table E-1, Section E.2 Table E-2, and Section E.3 Tables E-3a and E-3b) of the current version of 304-CD-005.	ICLHW
S-INS-61010	The ICLHW CI at the GSFC DAAC shall be capable of ingesting data from the EDOS at a maximum daily rate that is three times the nominal rate specified in Appendix E (Section E.1, Table E-1, Section E.2 Table E-2, and Section E.3 Tables E-3a and E-3b) of the current version of 304-CD-005.	ICLHW
S-INS-61020	The ICLHW CI at the LaRC DAAC shall be capable of ingesting data from the EDOS at the nominal daily rate specified in Appendix E (Section E.1, Table E-1, Section E.2 Table E-2, and Section E.3 Tables E-3a and E-3b) of the current version of 304-CD-005.	ICLHW

**Table A-1. Requirements Trace (17 of 17)**

<b>L4 Rqmt ID</b>	<b>L4 Requirement Text</b>	<b>Object Class, CSC, or CI</b>
S-INS-61025	The ICLHW CI at the LaRC DAAC shall be capable of ingesting data from the EDOS at a maximum daily rate that is three times the nominal rate specified in Appendix E (Section E.1, Table E-1, Section E.2 Table E-2, and Section E.3 Tables E-3a and E-3b) of the current version of 304-CD-005.	ICLHW
S-INS-61030	The ICLHW CI at the EDC DAAC shall be capable of ingesting data from the Landsat 7 Processing System (LPS) at the nominal rate specified in Appendix E (Section E.1, Table E-1, Section E.2 Table E-2, and Section E.3 Tables E-3a and E-3b) of the current version of 304-CD-005.	ICLHW
S-INS-61040	The ICLHW CI at the EDC DAAC shall be capable of ingesting data from the Landsat 7 IAS at the nominal daily rate specified in Appendix E (Section E.1, Table E-1 Section E.2 Table E-2, and Section E.3 Tables E-3a and E-3b) of the current version of 304-CD-005.	ICLHW
S-INS-61050	The ICLHW CI at the EDC DAAC shall be capable of ingesting data from the Landsat 7 IGSSs at the nominal daily rate specified in Appendix E (Section E.1, Table E-1, Section E.2 Table E-2, and Section E.3 Tables E-3 Tables E-3a and E-3b) of the current version of 304-CD-005.	ICLHW
S-INS-61080	The ICLHW CI at the GSFC DAAC shall be capable of ingesting data from the NMC at the nominal daily rate specified in Appendix E (Section E.1, Table E-1 and Section E.2 Table E-2, and Section E.3 Tables E-3a and E-3b) of the current version of 304-CD-005.	ICLHW
S-INS-61110	The ICLHW CI at the JPL DAAC shall be capable of ingesting data from RADAR-ALT at the nominal daily rate specified in Appendix E (Section E.1 Table E-1, Section E.2 Table E-2, and Section E.3 Tables E-3a and E-3b) of the current version of 304-CD-005.	ICLHW
S-INS-61115	The ICLHW CI at the JPL DAAC shall be capable of ingesting data from ADEOS II at the nominal daily rate specified in Appendix E (Section E.1 Table E-1, Section E.2 Table E-2, and Section E.3 Tables E-3a and E-3b) of the current version of 304-CD-005.	ICLHW
S-INS-61140	The ICLHW CI at the LaRC DAAC shall be capable of ingesting data from SAGE III at the nominal daily rate specified in Appendix E (Section E.1 Table E-1, Section E.2 Table E-2, and Section E.3 Tables E-3a and E-3b) of the current version of 304-CD-005.	ICLHW
S-INS-61150	The ICLHW CI at the ASF DAAC shall be capable of ingesting data from the ASF RGS at the nominal daily rate specified in Appendix E (Section E.1 Table E-1, Section E.2 Table E-2, and Section E.3 Tables E-3a and E-3b) of the current version of 304-CD-005.	ICLHW
S-INS-61160	The ICLHW CI at the ASF DAAC shall be capable of ingesting data from the ASF SPS at the nominal daily rate specified in Appendix E (Section E.1 Table E-1, Section E.2 Table E-2, and Section E.3 Tables E-3a and E-3b) of the current version of 304-CD-005.	ICLHW
S-INS-61170	The ICLHW CI at the LaRC DAAC shall be capable of ingesting data from ACRIM at the nominal daily rate specified in Appendix E (Section E.1 Table E-1, Section E.2 Table E-2, and Section E.3 Tables E-3a and E-3b) of the current version of 304-CD-005.	ICLHW

This page intentionally left blank.

## Appendix B. Program Design Language (PDL)

---

The Ingest Program Design Language (PDL) for non-trivial operations is included in this appendix. The PDL is sorted by object class and by object class operation. C++ syntax is used to identify object class operations (e.g., InBOMetadata::Preprocess indicates the Preprocess operation of the InBOMetadata object class). The PDL follows the standards set up in the PDL Program Instruction addendum to the Software Development Plan.

### **InBOMetadata::Preprocess**

```
Call InMetadataTool::PGS_MET_INIT to load target MCF into memory
Call InFile::Read to read metadata file into memory
DoWhile(Call InMetadataTool::GetNext=True)
    Call InSourceMCF::GetParInfo to obtain location of value associated with target
    parameter name
    Read value from appropriate location
    If (Data is Binary) Then
        Call ConvertBintoASCII to map binary expression to corresponding ASCII string via Look
        Up Table
    EndIf
    Call InMetadataTool::PGS_MET_SET to set the value of attribute in target MCF
End DoWhile
```

### **int InDAN::Check(char \*DAAmsgPtr)**

Open the DAA Error File; the file would contain one of the following code:

1. Accepted
2. Invalid DAN Sequence number
3. Invalid File Count
4. Invalid Data Service
5. Invalid Aggregate Length
6. Invalid Data Type
7. Invalid Directory
8. Invalid Time Stamp Format
9. Invalid Generation Time Format
10. Invalid File Size Field
11. Invalid Time/Date Format

Read the DAA Error code from the DAA file

Close the DAA Error File

if DAA Error code is 1-5

Construct the InShortDAA class

Call InShortDAA.FillDAA() to package the DAA message using DAA Error code

```

        and myDANSeqNo
else
    For each entry in myDataTypeList[]
        Assign DAA Error code to DAAStatus[i]
        Assign myDataTypeList[i].DataTypeId to DataType[i]
        Assign myDataTypeList[i].DatDescriptor to DataDescriptor[i]
    End for
    Construct the InLongDAA class
    Call InLongDAA.FillDAA() to package the DAA message using DAAStatus[],
        DataType[],DataDescriptor[], myDataTypeCount, and myDANSeqNo
endif
return

```

**InDAN::InDAN(char \*DANFile, char \*DAAMsgPtr, int status)**

```

Get the total byte size of DANFile
Allocate PVL-Buffer with total DANFile byte size
Open DANFile
Read the whole DANFile into PVL-Buffer
Close DANFile
Call ParsePVL() passing the PVL-Buffer address and the DANFile size to
    extract PVL keywords and Keyword-Values from the PVL-Buffer
Call Check() to verify the DAN components and fills the verification
    results in the DAA data message

```

**InDAN::InDAN(DANmsg \*DANmsgPtr, char \*DAAMsgPtr, int DAALength, int status)**

```

Calculate the total bytes of MsgHeader, EDU_Label, DAN_Label and
    assign to HdrLen
Set PVLptr to HdrLen byte after the first byte of DANmsgPtr
Extract the TotalMessageLength from MsgHeader
Calculate the PVLlen by subtracting TotalMessageLength from HdrLen
Call ParsePVL() passing the PVLptr and PVLlen
Call Check() to verify the DAN components and fills the verification
    results in the DAA message
return

```

**int InDAN::ExtractKeyword(char \*PVL\_stmt, char \*Keyword, char \*Keyword\_Value)**

```

*** Task complete by parser() ***
Search for 1st alphanumeric position in PVL_Stmt and assign to StartPtr
Set PVL_Stmt to Key_StartPtr
Search for 1st non-alphanumeric position in PVL_Stmt and assign to EndPtr
Extract StartPtr to EndPtr from PVL_Stmt and assign to Keyword
Call ExtractValue to extract the keyword value

```

return

**int InDAN::ExtractPVLStmt(char\* PVL\_Buffer, char \*PVL\_stmt, int PVL\_Stmt\_Len)**

\*\*\* Task complete by parser() \*\*\*

Search for 1st alphanumeric position in PVL\_Buffer and  
assign to StartPtr

Search for ';' in PVL\_Buffer and assign position to EndPtr and PVL\_Stmt\_Len

Extract PVL\_Stmt from PVL-Buffer starting StartPtr byte to Stmt\_EndPtr -1

return

**int InDAN::ExtractValue(char \*PVL\_stmt, char \*Keyword\_Value)**

\*\*\*Task complete by parser()\*\*\*

Search for 1st alphanumeric position after '=' in PVL\_stmt and assign  
position to StartPtr

Search for 1st non-alphanumeric position after StartPtr and assign  
position to EndPtr

Extract StartPtr to EndPtr from PVL\_Stmt and assign to Keyword-Value

return

**int InDAN::ExtractValue(char \*PVL\_Stmt, char \*Keyword\_Value)**

Task complete by parser()

return SUCCESS;

**int InDAN::FillData (char \*IngestType, char \*ParsedKeywords)**

Copy DAN information from daninfo data structure to the private  
member data structure.

return

**InDAN::CreateDANGranuleFiles (RWstring& DANfile)**

Creates DAN Granule files from the original DAN file. Each DAN Granule file will have sufficient header data to be considered a single granule DAN file.

Do for all granules in the DAN

Construct a DAN Granule File Name by adding the GranuleID to the original DAN file  
name

Open DAN file

Process open error, log, notify user, return

Open DAN Granule File

```

    Process open error, log, notify user, return
    If no open errors
        Read any DAN header PVL statements from DAN header which will be needed
    for
        remote processing of the DAN Granaule
        Process Read errors, log, notify user, return
        If no error
            Write needed header statements to DAN Granule File
            Process write errors, log, notify user, return
        Endif
        If no error
            Read Granule (File Group) PVL statements
            Process Read Error, log, notify user, return

        Endif
        If no error
            Write Granule PVL statements to DAN Granule File
            Process write error, log, notify user, return
        Endif
    Endif
Enddo
EndPDL

```

**int InDAN::ParsePVL(char\* PVL\_Buffer, int PVL\_Len)**

```

If PVL_Len is greater than 0 then call parse()
Call FillDAN() to put the extracted PVL statements into the
    InDAN class data memory
return

```

**int InDAN::GetDANSeq(void)**

```

return DANSeqNum

```

**InDataPreprocessTask::Preprocess**

```

BeginPDL
Call InDataTypeTemplate::GetDInfo to obtain a list of all required files
Do While(Call InDataPreprocessList::GetNext=True)
    Increment File Counter.. this is a RW list
    Store File Name
    Get file type of File Name
    Store File Type
    Increment appropriate file type counter
End Do While

```

```

If (required file types exist) Then
    Do While (Counter < number of files)
        Call InFileTypeTemplate::GetFTInfo to obtain file type information
        Instantiate appropriate preprocessing specialization
        If(File Type= Metadata)
            Instantiate DsCIDDescriptor
            Call DsCIDDescriptor::GetMCF to obtain target MCF
        Else
            EndIf
        End Do While
    Do While(Counter < number of files)
        Call appropriate preprocess operation
        Call InMetadataTool::PGS_MET_WRITE
        Instantiate a new file
        If(File Type=Metadata) Then

            Call DsCIDDescriptor::Validate
            If (Validate Fails) Then
                Log Errors
                Flag Metadata
            Else
                EndIf
        Else
            End If
    Enddo
Else
    Send Failure status back to InDataPreprocessTask
EndIf
EndPDL

```

### **InDataServerInsertionTask::SendInsert**

```

BeginPDL
Call DsCIESDTRreferenceCollector::DsCIESDTRreferenceCollector to set up Data Server session
Call DsCICCommand::DsCICCommand to instantiate command object and reference advertisement
Call DsCICCommand::SetParameters to include list of file types to be inserted
Call DsCIRequest::DsCIRequest to instantiate object and reference associated Command objects
Call DsCIRequest::Submit
Return
EndPDL

```

### **InDataTransferTask::TransferDataGranule (RWString DANGranuleFileName)**

```

BeginPDL
Construct InDan object from the InDANGranuleFile .. The InDANGranuleFile should look just
like a regular DAN file, but it will contain only one granule.
Clear DBHeader Error Flag

```

```

Allocate Staging Disk and Network Resources
If Failure
    Set DBHeader error flag
    Do for each file in DANGranuleFile
        Call InRequestFileInfo::UpdateFileStatus (ECSResourceAllocationFailure, Request-
ID,GranuleID, FileName)
    Enddo
Else
    Do for each file in the DAN Granule
        Construct InFile object
        Call InFile::Transfer
        If Error
            Set DBHeader error flag
            Call InRequestFileInfo::UpdateFileStatus
                (ECSFileTransferFailure, RequestID,GranuleID,
                FileName)
            Do for the rest of files in DAN Granule
                Call InRequestFileInfo::UpdateFileStatus
                (ECSFileTransferAbort, RequestID,GranuleID,
                FileName)
            Enddo
            Break out of upper do loop
        Else
            Call CheckGranuleState (ResultingAction)
            If (ResultingAction is "Cancel")
                Update Return status
                Return
            Else
                Add file to InDataPreprocessingList

                Call InRequestFileInfo::UpdateFileStatus
                (Success, RequestID,GranuleID,
                FileName)
            Endif
        Endif
    Enddo
    If DBHeader Error
        set EcUtStatus to Failed
    Else
        set EcUtStatus to Success
    Endif
    Return EcUtStatus

Endif
Return
EndPDL

```

### **InGranuleAsync\_C::Cancel**

This routine overloads EcSrAsynchReqeust\_C::Suspend. It will send a message to the InGranuleAsync\_S object to suspend remote processing.

BeginPDL

Call EcStAsynchRequest\_C::Cancel to notify the InGranuleAsync\_S object to suspend.

This is a synchronous call so when it returns, the object has suspended.

Delete granule from RWlist InGranuleList

Return EcUtStatus

EndPDL

### **InGranuleAsync\_C::Complete**

This routine overloads EcSrAsynchReqeust\_C::Complete. It will signal that a granule has completed processing.

BeginPDL

Delete granule from RWlist InGranuleList

Call EcStAsynchRequest\_C::Complete to perform generic completion activities

(Note all granule status will be available in the InRequestProcessData DB table when needed for DDN building)

Return EcUtStatus

EndPDL

### **InGranuleAsync\_C::InGranuleAsync\_C (InGranuleAsyncReqeust\_s& ctorMSG): EcCsAsynchReqeust\_C(ctorMSG)**

Constructor

BeginPDL

... inherits from EcCsAsynchReqeust\_C

set myServerUR to ctorMsg.GetUR():

EndPDL

### **InGranuleAsync\_C::~InGranuleAsync\_C (void)**

Destructor -- nothing to do here ---- EcSrAsynchRequest\_c should take care of everything

### **InGranuleAsync\_C::StateChange**

BeginPDL

When a state change notification request is received, appropriate resource utilization states are decremented.

Initialize state change count to 0 when the object is created

Increment StateChangeCount

```

If StateChangeCount equals 1
    Decrement network resource utilization
Elseif StateChangeCount equals 2
    Decrement processing utilization ( for the remote node)
Elseif StateChangeCount equals 3
    Decrement Archive Process statistics (for appropriate archive)
Else
    Process error message, log error
endif
Return
EndPDL

```

### **InGranuleAsync\_C::Suspend**

This routine overloads EcSrAsynchReqeust\_C::Suspend It will send a message to the InGranuleAsync\_S object to cancel (same as suspend to Granule Processing) remote processing . Note that a Resume function is not necessary, when a granule is resumed, InGranuleServer\_C will send a ResumeFlag with the ProcessGranule Message.

```

BeginPDL
Call InGranuleAsync_C::Cancel  Note that suspending a remote granule will actually cancel it.
Return EcUtStatus
EndPDL

```

### **InGranuleAsync\_S::Cancel**

```

BeginPDL
If EcSrAsynchReqeust_S myState is "Allocating Resources"
    Call InDataTransmitTask::CancelResource to Send Resource Manager Cancel Command
Elseif EcSrAsynchReqeust_S myState is "Inserting"
    Call InDataServerInsertionTask::SendCancel to Send DsClSubmit Cancel Command
endif
Call EcSrAsynchRequest_S::Cancel ... I am assuming the Default action here is to set myState to
Cancel
Return
EndPDL

```

### **InGranuleAsync\_S::SetPriority**

```

BeginPDL
If EcSrAsynchReqeust_S myState is "Allocating Resources" for disk space or transfer
    Call InDataTransmitTask::SetPriorityResources or SetPriorityTransferto Send Resource
Manager Set Priority Command
Elseif EcSrAsynchReqeust_S myState is "Inserting"
    Call InDataServerInsertionTask::SendPriority to Send DsClSubmit Set Priority Command
endif

```

```

Call EcSRAsynchRequest_S::SetPriority
Return
EndPDL
InGranuleAsync_S::Suspend

```

Note:: A cancel is sent to Resources manager if resources are being allocated, not a suspend

```

If EcSrAsynchRequest_S myState is "Allocating Resources"
    Call InDataTransferTask::Cancel to Send Resource Manager Cancel Command
Elseif EcSrAsynchRequest_S myState is "Inserting"
    Call InDataServerInsertionTask::SendSuspend to Send DsClSubmit Suspend Command
endif

Call EcSRAsynchRequest_S::Suspend

```

### **InGranuleAsync\_S::CheckGranuleState (EcTInt RequestID, EcTInt GranuleID, RW-String ResultingAction)**

This routine will check to see if a granule state change has occurred. ResultingActions will be returned to the caller to provide "what next" processing directions.

```

    Get GraunuleState from EcSrAsynchRequest_S::GetState
    Process error status returned from a RequestState function call, log error, notify operator, set
error          status and return

    If ReqeustState is not equal to active
        If GranuleState is equal to "cancel"
            Set ResultingAction to "Cancel"
        ElseIf GranuleState is equal to "suspend"
            Set ResultingAction to "Cancel" suspend and cancel are handled the same way
        Else ... error state Note: resume/recover is not sent to a granule and set priority is han-
dled          totally by our overloaded InGranuleAsync_S::SetPriority function.
            Notify operator of error ... invalid Request state is specified
            Log error
            Return
        Endif
    Endif
Return
EndPDL

```

### **InGranuleAsync\_S::Execute**

This routine will process the Granule. Processing steps include 1. Transferring all files referenced in the DanGranuleFileName file, 2. Preprocessing all transferred files, 3. Inserting Granule into the DataServer for Archiving

```

Construct InDataTransferTask Object (myDANGranuleFileName, ResumeFlag, DataTypeID,
TransferredFileList)
Call InDataTransferTask::TransferData (RequestID, GranuleID)
If good status
    Call CheckGranuleState (RequestID, GranuleID, ResultingAction)
    If ResultingAction is "Cancel"
        Call InGranuleAsync_S::Cancel
            to cleanup and also to notify client of completion of granule
    return
Endif
Call EcSrAsynchRequest_C::StateChange ... this call will notify InGranuleAsync_C that
the 1st state
    change (files have been transferred) has occurred
    InGranuleAsync_C will decrement network resource
    utilization stats
Construct InDataPreprocessTask(DataTypeID, &preprocessList) object

Call InDataPreprocessTask ()
Process returned error status
If good status
    Call InRequestProcessData::UpdateGranuleState(RequestID,GranuleId) to "pre-
processed"
    Call CheckGranuleState (RequestID,GranuleID, ResultingAction)
    Process error status
    If Good Status
        If ResultingAction is "Cancel"
            Call InGranuleAsync_S::Cancel
                to cleanup and also to notify client of granule cancellation
        return
    Endif
    Call EcSrAsynchRequest_C::StateChange ... this call will notify
InGranuleAsync_C that the 2nd state
    change (files have been preprocessed has occurred)
    InGranuleAsync_C will decrement resource
    utilization stats

    Construct InDataServerInsertion Task object InsertObject
    Call InDataInsertionTask::SendInsert ( InsertObject)
    Process Error Status
    If Good Status
        Call InRequestProcessData::UpdateGranuleState
            (RequestID,GranuleID, "Archived")
        Call EcSrAsynchRequest_C::StateChange ... this call will notify
InGranuleAsync_C that the 3rd state
    change (files have been archived) has occurred
    InGranuleAsync_C will decrement resource

```

```

        utilization stats
    Else ....SendInsert Error
        Call InRequestProcessData: UpdateGranuleState
            (RequestId,GranuleID, "ArchiveFailed")
    Endif
Else ...Preprocess Error
    Call InRequestProcessData: UpdateGranuleState
        (RequestId,GranuleID,"PreprocessFailed")
    Endif
Else ... Transfer Error
    Call InRequestProcessData: UpdateGranuleState
        (RequestId,GranuleID, "TransferFailed")
    Endif
Endif
EndPDL
InGranuleAsync_S::InGranuleAsync_S(RWString &DANGranuleFileName, EcTBoolean ResumeFlag, EcTInt RequestID, EcTInt GranuleID, InGranuleAsync_c*& async)

BeginPDL
Set myDANGranuleFileName
Set myResumeFlag
Set myClientUR
Set myRequestID
Set myGranuleID
EndPDL

InGranuleAsync_S::~~InGranuleAsync_S

Destructor

InGranuleServer_S::InGranuleServer_S(EcURUr myUR, EcSrRequestDispatcher* localServer)

Constructor

InGranuleServer_S::ProcessRequest (RWString &DANGranuleFileName, EcTBoolean ResumeFlag, EcTInt RequestID, EcTInt GranuleID, InGranuleAsync_c*& async) , returns InGranuleAsync_C*

BeginPDL
Create a new request InGranuleAsync_S from passed in attributes
Add the created request to the request queue
Create new client request by creating a new InGranuleAsync_C( new InGranuleAsync_S request)
Return (new client request) ... to ReceiveMsgRAcceptance
EndPDL

```

**InGranuleServer\_C::InGranuleServer\_C(EcUrUR serverUR):EcCsRequestServer\_C(serverUR)**

BeginPDL

Connect() ... This function is inherited from EcCsReqeustServer\_C

EndPDL

**InGranuleServer\_C::~InGranuleServer\_C(void)**

BeginPDL

Disconnect(): ... This function is inherited from EcCsReqeustServer\_C

EndPDL

**InGranuleServer\_C::ProcessGranule(RWString &DANGranuleFileName, EcTBoolean ResumeFlag, EcTInt RequestID, EcTInt GranuleID, InGranuleAsync\_c\*& async)**

This routine builds and send Granule Process request message to the remote server

Note RequestID and GranuleID are passed so that the remote server can update the data base to checkpoint state information

BeginPDL

Create ProcessGranule Message (i.e new ProcessGranuleMsg (DANGranuleFileName, ResumeFlag) .. (Note:: the DANGranuleFileName is the result of calling InDAN::CreateDANGranuleFiles,

The ResumeFlag is true if a granule is being resumed, otherwise false. Note that when a granule is suspended, remote async objects are deleted)

Send Process Granule message to the server via the EcMhMsgHandler::SendMsgRAcceptance (myServerUR, ProcessGranuleMessage, (EcUtStreamable\*&)async)

If status is good

Add InGranuleAsync\_C reference to RWlist InGranuleList, set Granule Status to "Active"

Return (EcUtStatus)

EndPDL

**InGranuleServer\_S::Main**

This is the driver module for InGranuleServer\_S

BeginPDL

init Process Framework , registrar metric files, get configuration parameter?,

Call PFStart

End  
EndPDL

**InGranuleServer\_S::ReceiveMsgRAcceptance( EcCsMsg\* newMessage, EcURUr\* clientUR);**

Receives requests sent from InGranuleServer\_C client software to process the Granule.

BeginPDL

Instantiate an InGranuleAsync\_C object

Extract message class ID from the new message

Based on ID, verify the message received is a Process Granule message .. the only type a message we expect on the server level

Get DAN Granule file name from the message

Get ResumeFlag from the message..

Get RequestID from the message.

GetGranuleID from the message.

Call InGranuleServer\_S::ProcessGranule to initiate the processing of the Granule by InGranuleAsync\_C,

Returns (EcUtStreamable\*) theInGranuleAsync\_C.

EndPDL

**InHDFMetadata::Preprocess()**

BeginPDL

Call InMetadataTool::PGS\_MET\_INIT to read target MCF into memory

Call appropriate HDF I/O tools to read file metadata

Identify HDF metadata object

DoWhile(Call InMetadataTool::GetNext=True)

    Compare target parameter name with file string

    Call InMetadataTool::PGS\_MET\_SET to set the value of attribute in target MCF

End DoWhile

EndPDL

**InInteractiveIngest::ProcessUserName (RWString UserName)**

BeginPDL

Get UserName and validate

Report validation error to user

Return

EndPDL

### **InInteractiveIngest::GetDANHeader(RWList HeaderInformation)**

BeginPDL

Get DAN Header information from Form (originating system, consumer system, sequence number, expiration time if applicable)

Call InDAN::ValidateHeader .. to validate

If error

    notify user, Log error

Endif

Return

EndPDL

### **InInteractiveIngest::GetDataTypeInformation( RWList DataTypeInformation)**

BeginPDL

Get DAN Data Type information from Form (data type, data descriptor, data version)

Call InDAN::ValidateDataType.. to validate

If no error

    If Data type is Scanned\_document

        If Operator has not provided Associated Metadata information

            Instruct operator to fill outHTML Document Metadata form

            Return

        Endif

    Endif

Else

    Notify user of error, log error

Endif

Return

EndPDL

### **InInteractiveIngest::ProcessDocumentMetadata**

BeginPDL

Get Document Metadata from Form

Validate

Notify user of validation errors, log error

Return

EndPDL

**InInteractiveIngest::ProcessDirSpecSingleDAN (RWString DirectoryName, RWString  
DataType, RWList DANHeader)**

BeginPDL

Get Directory specification

Call GetDataTypeInfo (RWList DataType)

Call CreateFileList

Do for all files in the FileList

    Display file name to user and request file type

    Call DetermineFileSize

    Process error, display to user, continue at Do

Enddo

Call BuildDAN(FileList,DataTypeList,DANHeader)

If no errors and "BuildDAN and Ingest" option selected

    Call SendRequest .... to send Ingest Request to InRequestManager

Else

    Report error to user, log errors

Endif

EndPDL

**InInteractiveIngest::ProcessDirSpecMultiDAN (RWString DirectoryName, RWString  
DataType, RWList DANHeader)**

BeginPDL

Get Directory specification

Call CreateFileList

Do for all files in List

    Call GetDataTypeInfo (RWList DataType)

    Display file name to user and request file type

    Call DetermineFileSize

    Process error, display to user, continue at Do

    Call BuildDDN (FileList,DataTypeList,DANHeader)

    If no errors and "BuildDAN and Ingest" option selected

        Call SendRequest .... to send Ingest Request to InRequestManager

    Endif

Enddo

Return

EndPDL

**InInteractiveIngest::InBuildDAN (RWList FileList, RWList DataType, RWListDANHead-  
er)**

```

BeginPDL
Open file
If no error
    Call WriteDANHeader
    If no error
        Call WriteDANGroup(RWList FileList, RWList DataType)
    else
        Process error.. notify user, log return
    endif
else
    Process error.. notify user, log return
endif
EndPDL

```

### **InInteractiveIngest::WriteDANGroup(RWList FileList, RWList DataType)**

```

BeginPDL
Write PVLGroup indicator
Process error.. Notify user, log, return
Write Data Type Information in PVL form
Process error.. Notify user.. log,return
Do for all files
    Write file spec information
    Process error.. notify user ,log, return
    Write end of file group
Enddo
EndPDL

```

### **InInteractiveIngest::ProcessInteractiveDANSpec (void)**

```

BeginPDL
Call GetDANHeaderInformation
Process errors.. notify user, log go back to Call GetDANHeaderInformation
Call WriteDANHeader
Process errors.. notify user, log return

If valid
    Get user option to 1) Build DANand Ingest, or 2) Build DAN only ... DAN could then be
    used at a later date for hard media ingest
Do while not DAN complete
    Get and Validate DataTypeID
    If valid
        Do while not DataSetComplete
            Get FileName
            Get FileLocation

```

```

        Get user option to 1) Supply file size interactively, or 2) request Ingest to
access        file directory to determine file size
        If Ingest should determine file size
            Call DetermineFileSize
            Process error - display error to user
            continue ... go back to do
        Endif
    Enddo
    Call WriteDANGroup
Endif
Enddo ... DAN complete

If no errors and "BuildDAN and Ingest" option selected

    Call SendRequest .... to send Ingest Request to InRequestManager

Endif

Return EcUtStatus

EndPDL

```

### **InInteractiveIngest::InDetermineFileSize(RWString FileName, RWString FileLocation)**

This routine requires that ECS have network access to the file being processed

```

Begin PDL
If Ingest should determine file size
    Call appropriate UNIX utilities to determine file size
    Process errors.. notify user, log

    If Error
        Return EcUtStatus
    Endif

Else
    Get file size from Form
    Process AtoInt errors
Endif

EndPDL

```

### **InInteractiveIngest::ProcessDANIngest (void)**

```

Begin PDL

Get DAN file name
Call InDAN::InDAN to construct and validate DAN object

```

```

Process error, notify user,log, return
If no error
    Call InDAN::Checkpoint .. to checkpoint DAN file into the Ingest DB
    If no error
        Send RPC to request Manager to proccess DAN Ingest Request
    Else
        Notify user,log
        Return
    Endif
Else
    notify user,log
    return
Endif
Return EcUtStatus
EndPDL

```

**InInteractiveIngest::ProcessStatusRequest RWString UserName)**

```

BeginPDL

Call InRequestProcessHeader::GetInteractiveUsersStatus (UserName)
Process error , notify user of error, log, return
Display request status
Return

EndPDL

```

**int InLongDAA::FillDAA (int DAAStatus[], char \*DataType[], char \*DataDescriptor[],  
int DataTypeCount, int DANSeqNo)**

```

If (DANSeqNO is greater than 0)
    Assign DANSeqNo to myLongDAA.DANSequenceNum
    Assign DataTypeCount to myLongDAA.FileGroupCount
    Init MsgLen to LongDAAMsgHeader
    For i = 1 to DataTypeCount
        Assign DAAStatus[i] to myLongDAA.FileGroup[i].Disposition
        Assign DataType[i] to myLongDAA.FileGroup[i].DataType
        Assign DataDescriptor[i] to myLongDAA.FileGroup[i].Descriptor
        Increment MsgLen by the string length of DataType[i]
        and DataDescriptor[i]
    End for
Else
    write invalid DAN Sequence Number into the event log
End if
Assign the least significant 3 bytes of MsgLen to myLongDAA.MsgLength
return

```

**int InLongDAA::GetDAA(LongDAAMsg \*DAAMsg, int DAAMsgLen)**

Assign myLongDAA to DAAMsg  
Call InMessage::GetMsgLength() and assign to DAAMsgLen  
If (DDNmsgLen is <= zero)  
    write Empty DAA data message error into the event log  
Endif  
return

**InLongDAA::InLongDAA()**

Assign the lease significant byte of InCLongDAAType to myLongDAA.MsgType  
return

**int InLongDAA::FillDAA (int DAAStatus[], char \*DataType[], char \*DataDescriptor[],  
                          int DataTypeCount, int DANSeqNo)**

If (DANSeqNO is greater than 0)  
    Assign DANSeqNo to myLongDAA.DANSequenceNum  
    Assign DataTypeCount to myLongDAA.FileGroupCount  
    Init MsgLen to LongDAAMsgHeader  
    For i = 1 to DataTypeCount  
        Assign DAAStatus[i] to myLongDAA.FileGroup[i].Disposition  
        Assign DataType[i] to myLongDAA.FileGroup[i].DataType  
        Assign DataDescriptor[i] to myLongDAA.FileGroup[i].Descriptor  
        Increment MsgLen by the string length of DataType[i]  
        and DataDescriptor[i]  
    End for  
Else  
    write invalid DAN Sequence Number into the event log  
End if  
Assign the least significant 3 bytes of MsgLen to myLongDAA.MsgLength  
return

**int InLongDAA::GetDAA(LongDAAMsg \*DAAMsg, int DAAMsgLen)**

Assign myLongDAA to DAAMsg  
Call InMessage::GetMsgLength() and assign to DAAMsgLen  
If (DDNmsgLen is <= zero)  
    write Empty DAA data message error into the event log  
Endif  
return

**InLongDAA::InLongDAA()**

Assign the lease significant byte of InCLongDAAType to myLongDAA.MsgType  
return

**int InLongDDN::FillDDN (int DDNStatus, char \*FileDir[], char \*FileId[],  
int FileCount, int DANSeqNo)**

If (DANSeqNo is greater than 0)  
    Assign DANSeqNo to myLongDDN.DANSeqNo  
    Assign FileCount to myLongDDN.FileCount  
    Init MsgLen to byte size of LongDDNmsgHeader  
    For i = 1 to FileCount  
        Assign DDNStatus[i] to myLongDDN.File[i].Disposition  
        Assign FileDir[i] to myLongDDN.File[i].Directory  
        Assign FileId[i] to myLongDDN.File[i].FileName  
        Increment MsgLen by the string length of FileDir[i] and FileId[i]  
    End for  
Else  
    write invalid DAN Sequence Number into the event log  
End if  
Assign the least significant 3 bytes of MsgLen to myLongDDN.MsgLength  
return

**int InLongDDN::GetDDN(LongDDNmsg \*DDNmsg, int DDNmsgLen)**

Assign myLongDDN to DDNmsg  
Call InMessage::GetMsgLength() and assign to DDNmsgLen  
If (DDNmsgLen is <= zero)  
    write Empty DAA data message error into the event log  
Endif  
return

**InLongDDN::InLongDDN()**

Assign the lease significant byte of InCLongDDNType to myLongDDN.myMsgType  
Return

**InMediaIngest::GetBarCode(RWString BarCode)**

Note: I have not seen the requirement for Bar Code readers yet!

BeginPDL  
Call DsStResourceProvider::GetBarCode(RWString BarCode)  
Process error

```
If no error
    Display BarCode on GUI
Endif
```

```
Return EcUtStatus
EndPDL
```

### **InMediaIngest::ProcessRequest(void)**

```
BeginPDL
```

```
Get Tape ID input method from GUI Screen
```

```
If method is BarCode
```

```
    Call DsStResourceProvider::GetBarCodeReaderResource
```

```
    Process Error status
```

```
    If no error
```

```
        update TapeIDMethod to indicate BarCode
```

```
    Else
```

```
        Return EcUtStatus Failed
```

```
    Endif
```

```
Else
```

```
    Update TapeIDMethod to indicate user supplied
```

```
Endif
```

```
Set EcUtStatus to Success
```

```
EndPDL
```

### **int InMessage::GetMsgLength(char \*MsgPtr)**

```
If (MsgPtr is not NULL)
```

```
    Move 2nd to 4th byte from MsgPtr and assign to MsgLen
```

```
else
```

```
    Assign zero to MsgLen
```

```
endif
```

```
return MsgLen
```

### **InMessage::InMessage()**

```
Return
```

### **InPollingIngestSession::ProcessRequest**

```
BeginPDL
```

```
Processing wakes up on polling intervals
```

```
InDAN::CheckpointRequest
```

```

Access MSS configuration files ::GetPriority to get the current request priority
Process error by notifying operator and updating log
Call InExternalDataProviderThreshold::GetState to get the current state of the external Data Pro-
vider
Process errors by writing to log .. continue with request
Process directory
Build DAN
If State is equal to suspend
    set DAA status to indicate the request has failed since ECS has suspended the External
DataProviders Session
    Set request failure status
    return
Else
    Send DAN request to InRequestManager
Endif
Go wait for next polling interval
EndPDL

```

### **InPVMetadata::Preprocess()**

```

Call InMetadataTool::PGS_MET_INIT to load target MCF into memory
DoWhile(Call InMetadataTool::GetNext=True)
    Call InSourceMCF::GetParInfo to match target parameter name with source parameter
name
    Call InFile::Read to read metadata file into memory
    DoUntil(Source Parameter=File String)
        Compare Source Parameter with File String
    End Do Until
    Extract value from parameter -value statement
    Call InMetadataTool::PGS_MET_SET to set the value of attribute in target MCF
End DoWhile

```

### **InRequest::Cancel (RWString ResultingAction)**

```

Begin PDL
Call InRequest.NotifyRemoteGranules(RequestID, MsgType) .. to cancel remote granule process-
ing
If returned status is bad
    Log error, set "detail", so that a single message will go to operator when request cancel-
lation is complete
    Return
Endif
Set ResultingAction to "Cancel"
Return status
End PDL

```

### **InRequest::CheckRequestState( EcTInt \*GranuleID, EcTInt ResultingAction)**

This routine will check to see if a request state change has occurred. If a state change has occurred, the routine will call the appropriate functions to perform the change. ResultingActions will be returned to the caller to provide "what next" processing directions.

```
Get request state from InReqeust object ::GetRequestState
If RequestState is not equal to active
    If RequestState is equal to "cancel"
        Call InRequest::Cancel(RequestID, ResultingAction), return status
    ElseIf RequestStat is equal to "suspend"
        Call InRequest::Suspend (RequestID, ResultingAction), return status
    ElseIF ReqeustState is equal to "set priority"
        Call InRequest::SetPriority(RequestID, ResultingAction), return status
    ElseIf RequestState is equal to "resume" ... Note that Resume and Recover act on a sin-
        granule whereas the other RequestState func-
        tions will
            affect all granules in the request
            Call InRequest::Resume(RequestID,GranuleID, ResultingAction), return status

    ElseIf RequestState is equal to "recover"
        Call InRequest::Recover(RequestID,GranuleID, ResultingAction), return status

    Else
        Notify operator of error ... invalid Request state is specified
        Log error
    Endif
    Process error status returned from a RequestState function call
Endif
Endif
Return
EndPDL
```

### **InRequest::CheckVolumeThreshold (EctInt GranuleID,RWString ResultingAction)**

This function will verify that that request is not exceeding volume threshold (assigned by InRequestManager when the request was started) If thresholds will be exceeded, the routine will suspend processing for n seconds (n is an MSS configuration parameter) then recheck thresholds. If during the suspend, previously sent granules have released needed resources (network, cpu, archive), the request processing will continue, otherwise processing will suspend again.

BeginPDL

Do until processing can continue

Call InRequest:: CheckRequestState ... to check for cancel or suspend /resume set priority control commands

```

        If ResultingAction is set to "Continue"
            Check granule volume variables in InGranuleServer_C object (singleton). Volume vari-
ables            include: CurrentNetworkVolume, CurrentPreProcVolume, CurrentIn-
sertVolume
            (Note: as Remote granule processing changes state (from transferring, to prepro-
cessing to      inserting, InGranuleAsync_C will be notified so that current volume
variables can be updated.)
            If when this granule is activated, granule volume variables will exceed threshold allo-
cation
                Sleep n seconds
            else
                Set processing can continue flag
            Endif
        Else
            Return
        Endif
    Enddo
Return
EndPDL

```

### **InRequest::CompleteRequestProcessing (EcTString ResultingAction)**

This routine completes the request, ResultingAction can be set to Suspend, Cancel or Complete

Call InrequestProcessHeader::DetermineHeader... to determine if request completed successfully or with errors and to build the DDN file or buffer. (DetermineHeader calls a stored procedure to query InRequestProcessData and return a buffer of file info and file status). The stored procedure will also update the InRequestProcessHeader with request status and request time statistics.

Process returned buffer to determine is a Short or Long DDN should be built.

Build short or long DDN

```

If ingest type is automatic network
    Call DeleteAutomatedNetwork .. pass DDN pointer
Elseif ingest type is hard media
    Call DeleteHardMedia .. pass in DDN pointer
Elseif ingest type is interactive
    Call DeleteInteractive .. pass in DDN pointer
Elseif ingest type is polling
    Call DeletePollingIngest .. pass in DDN pointer
else
    Process error, notify operator, log
endif
Return
EndPDL

```

### **InRequest::DeleteInteractiveIngest**

Access user profile to determine status directory locations

Process error, notify operator and log, return

If Archive was successful

    Deallocate staging

    Notify operator of error, log

    Delete Original files from user's directory

    Notify operator of error, log

Else

    Deallocate staging

    Notify operator of error, log

    Rename Original file to user's error directory (if necessary cleanup user's error directory before rename)

    Notify operator of error , log

Endif

Notify user via EMAIL of Ingest status (DDN file contents reported in Email MSG)

Process error, notify operator and log

Store DDN file in users's DDN directory using data type and date for file name  
(the user will be able to view this file via the HTML interface)

Process error, notify operator and log

Return

### **InRequest::GetGranuleServerUR(InGranuleServer\_C&, RWstring& GranuleDataType)**

BeginPDL

    Call InDataTypeTemplate:: GetGranuleServerUR(GranuleDataType)

    ...To get UR for the GranuleServer\_S which will process the GranuleDataType

    Process error status

    Search RWlist containing GranuleServer\_S URs which have already been processed through this routine and therefor have existing InGranuleServer\_C objects

    If the needed UR is not in the RW list

        Construct InGranuleServer\_C (GranuleServerUR) object for the GranuleServer\_S found in the RW list... this is the client object

    Else

        Extract InGranuleServer\_C Client object identifier from the RW list

    Endif

    Pass back InGranuleServer\_C &

    Return EcUtStatus

EndPDL

### **int InRequest::InRequest (DANmsg \*DANmsgPtr)**

Call InRequest::Check () to parse the PVL contents and check their validity

If (check is successful)

```

    Fill InRequest attributes
    Call InRequestProcessHeader::InRequestProcessHeader to checkpoint request processing at-
        tributes
    Call InRequestProcessData::InRequestProcessData to checkpoint request data type processing
        attributes
    Call InRequestFileInfo::InRequestFileInfo to checkpoint request file processing attributes
    Call InRequestSummaryHeader::InRequestSummaryHeader to checkpoint request summary
        attributes
    Call InRequestSummaryData::InRequestSummaryData to checkpoint data type summary at-
        tributes
    Return the object pointer (OID) for this object
Else
    Return error status
endif

```

### **InRequest::InRequest (char \*DANFile)**

```

Construct InDAN class using pointer to the DAN file
If (InDAN construction indicates success)
    Call InDAN::FillDAN() to read the DAN contents
    Call InRequest::Check () to parse the PVL contents and check their validity
    If (check is successful)
        Fill InRequest attributes
        Call InRequestProcessHeader::InRequestProcessHeader to checkpoint request processing
            attributes
        Call InRequestProcessData::InRequestProcessData to checkpoint request data type pro-
            cessing attributes
        Call InRequestFileInfo::InRequestFileInfo to checkpoint request file processing attributes
        Call InRequestSummaryHeader::InRequestSummaryHeader to checkpoint request sum-
            mary attributes
        Call InRequestSummaryData::InRequestSummaryData to checkpoint data type summary
            attributes
        Return the object pointer (OID) for this object
    Else
        Return error status
    endif
Else
    Return error status
end if

```

### **InRequest::ProcessRequest (EcTVoid)**

Initiates the ingest processing. ProcessRequest consist of calling InGranuleServer\_C for each Data Type group (granule) in the request (DAN). InGranuleServer\_C will send the granule request to InGranuleServer\_S which in turn creates an InGranuleAsync\_C and InGranuleAysnc\_S

to handle actual granule processing. ProcessRequest as it loops over all data types, will call CheckVolumeThreshold. to check for resource shortages. Once all granules have been sent off for processing, ProcessRequest will wait for notification that all granules have been processed. During this wait, ProcessRequest would continue to monitor any changes to request state. When all granules are completed, ProcessRequest will call Delete to terminate request processing.

BeginPDL

Construct InDAN:object.

Call InDAN::CreateDANGranuleFiles... This routine will partition the DAN File into separated DANGranule files where the Granule files contains all PVL statements needed for remote processing of the granule

Get DataTypeList (Granule list) from MyDANfile ... note DataTypeList contains the data type Id and the DANGranuleFile ID

For Each Data Type group entry

Call CheckVolumeThreshold (RequestID, GranuleID, Resulting \_Action) .. checks for and processes suspend, recover/resume, or cancel control commands. Also determines if request processing should be temporarily suspended due to resource shortages. CheckRequestState will perform the temporary suspension if it is necessary.

Process error status returned from CheckVolumeThreshold call

If ( ResultingAction is "SkipGranule" .. recovery/resume has been requested and this granule has already been completed

break - "go to next granule"

elseif (ResultingAction is "Cancel or "Suspend... cancel or suspend has been requested

Call CompleteRequestProcessing .. passing in cancel, suspend or complete

return

Else

Log error, notify operator, return

return

Endif

...Continue with processing

Call GetGranuleServerUR(myGranuleServer\_C&, InDataTypeList(I).DANGranuleFileName)

... Note several data type granules can be serviced by a single server so this routine will check first to see if the server is already

exists. If not it will construct the server. An InGranuleServer\_C object will be passed back, EcUtStatus will be returned

Process error: log error, notify operator, break to continue with next granule

Call myGranuleServer\_C.ProcessGranule ... to start asynchronous processing of the data type granule

```

        Process error: log error, notify operator, continue with next granule
    End For
    Call WaitForCompletion
    Return
EndPDL

```

### **InRequest::NotifyRemoteGranules (EctInt MsgType)**

Based on MsgType, the appropriate InGranuleAsync\_C function will be called. Note that all of these functions are synchronous, so once this routine has completed, remote granule processing has been notified.

Begin PDL

Do for all InGranuleList This is a RW list of pointers to all processing or completed granules, The list also contains each granules state

```

    If granule is not complete or has not failed....
        Set GranuleObj to Get InGranuleAsync_C address from RW list

        If MsgType is "Suspend"
            Call InGranuleAsync_C::Suspend
        elseif MsgType is "Cancel"
            Call InGranuleAsync_C::Cancel
        Elseif MsgType is "SetPriority"
            Call InGranuleAsync_C::SetPriority
        Else
            Log error ... invalid MsgType received
            Notify Operator
            Retrun
        Endif
        Construct GranuleRequestMsg.. set type to MsgType
        Send InGranuleRequestServer_C::SendMsg ( GranuleRequestMsg )
    Endif

```

```

Enddo
Return
EndPDL

```

### **InRequest::Recover(EcTInt GranuleID, RWString ResultingAction)**

Begin PDL

```

Call InRequestProcessData.GetGranuleStatefor Recover (RequestId,GranuleID,ResultingAction)
    ..to determine if the granule was completed prior to the request suspend or request abort or
if a resume has previously been attempted on the granule. This stored procedure will
    also set myRetryFlag to 1
Return status from InRequestProcessData call ... ResultingAction is also set there

```

End PDL

### **InRequest::Resume(EcTInt GranuleID, EcTInt ResultingAction)**

Begin PDL

Call InRequestProcessData.GetGranuleState .. to determine if the granule was completed prior to the request suspend.

If returned status is bad or indicates granule not found

    log error appropriate error

    Set EcUtStatus to Failed Granule

    Return

Endif

If granule was completed

    Set ResultingAction to "SkipGranule"

Else

    Set ResultingAction to "Continue"

Endif

Return good status

End PDL

### **InRequest::SetPriority(RWString ResultingAction)**

Begin PDL

Call InRequest.NotifyRemoteGranules(MsgType) .. to SetPriority of remote granule processing

If returned status is bad

    Log error

    Notify operator of failure

    Return

Endif

Set ResultingAction to "continue"

Return status

End PDL

### **InRequest::Suspend(RWString ResultingAction)**

Begin PDL

Call InRequest.NotifyRemoteGranules(RequestID, MsgType) .. to Suspend remote granule processing

If returned status is bad

    Log error, set "detail", so that a single message will go to operator when request cancellation is complete

    Return

```

Endif
Set ResultingAction to "Suspend"
Return status
End PDL

```

### **InRequest::WaitForCompletion**

```

BeginPDL
    Wait for CondSignal from granule pthread which will indicate all granules in request have
    completed      processing or been suspended or cancelled
    Call CheckRequestState (RequestID,GranuleID, ResultingAction)
    Process error status returned for CheckRequestState by logging error, notify operator, set
    request      status to complete, continue
    Call CompleteRequestProcessing .. passing in cancel, suspend or complete

Return
EndPDL

```

### **InRequestController::CreateRequestList (RWString ControlCommand, RWList Request-List)**

This routine will determine if the request is for 1. a single request, 2. all requests for a single data provider,  
3. allrequests

```

BeginPDL
If ((suspend or resume or set priority) and (all requests for a single data provider or all requests))
    Do for all External Data Providers selected
        If suspend or resume
            Call InExternalDataProviders::UpdateState to update flag to Session that that its
session      is suspended/resumed
        else
            Call InExternalDataProvider::UpdatePriority
        Endif

        Process error status (log, and notify operator)
    Enddo
Endif
Create list of requests to modify
Return
EndPDL

```

### **InRequestController::ProcessRequestStateChange(EcTInt RequestID, RWString State-ControlCommand)**

This routine is activated when the operator issues a request control command from the Monitor and Control GUI window

Begin PDL:

Based on State Control Command received, create list of requests to be control (this could be a single request, all request for a single data provider or all requests in the system)

Call CreateRequestList (ControlCommand, RequestList&)

Do for each request in the request list

    Call InRequestProcessHeader::UpdateStateForStateChange (RequestID, StateControl-Command, NewPriority, PreviousState)

        If Detail error reference in EcUtStatus indicates failure

            Failed to locate in DB

            InvalidState change request (previous state was not compatible with new state)

            Invalid request since previous state change has not completed

            Other

            Report appropriate error message to operator

            Log error message

            Return

        Endif

        If Previous State is TapeSession and State ControlCommand is not "resume"

            Display operator message indicating suspend/cancel/setpriority should be entered via status box on initiating GUI

            Return

        Endif

    Send RPC to InRequestManager to suspend/cancel/resume/setpriority.

Enddo .. over all requests

**InRequestInfo::AddRequest(int DANSeqNum, int RequestId)**

Check for duplicate DAN

If duplicate DAN not found

    Allocate memory for the new entry for the RequestInfo List

    Fill the new entry with RequestInfo

    Increment the Request count

EndIf

return

**InRequestInfo::DeleteRequest(int DANSequenceNum)**

set FOUND to false

```

for each RequestInfo in the RequestInfo List
    if RequestInfo[i].DANSequenceNum = DANSequenceNum
        set FOUND to true
        exit the loop
    endif
end for
if FOUND is true
    delete i-th RequestInfo entry from the list
    decrement the RequestInfo count
else
    return no match
endif
return

```

### **InRequestInfo::GetRequestCount(int \*RequestCount)**

Get the total number of requests from the list

### **InRequestInfo::InRequestInfo()**

Initialize the request list

### **InRequestInfo::InRequestInfo()**

Initialize the request list

### **InRequestInfo::~~InRequestInfo()**

Delete any dynamically allocated memory

### **InRequestInfo::ListRequests()**

```

For each request in the list
    print DANSeqNum and RequestId
END for
Print Total number of requests
return

```

### **InRequestInfo::operator==(const InRequestInfo &r)**

Defines an element in the RequestInfo list to be equal  
only if the DANSeqNum and RequestId match.

### **InRequestInfo::SearchRequest(int DANSequenceNum)**

```

Set Found to FALSE
For each entry in the RequestInfo List
    if RequestInfo.DANSequenceNUm = DANSequenceNum
        set FOUND to TRUE
        exit loop
    endif
endfor
return

```

#### **int InRequestList::AddRequest(int \*RequestID)**

```

Add RequestID to list based on priority
Checkpoint RequestList to data base
return

```

#### **int InRequestList::DeleteRequest(int \*RequestID)**

```

Delete RequestID from list
Checkpoint RequestList to data base
return

```

#### **int InRequestList::SearchRequest(int \*RequestID)**

```

Find entry with specified RequestID in list
return

```

#### **int InRequestManager\_ C::CancelRequest(int\* RequestID)**

```

Call InRequest::Cancel () to cancel the request at its current state
return

```

#### **DCEObjRefT\* InRequestManager\_ C::CreateRequest(DANmsg\* DANmsgPtr)**

```

Invoke the InRequestManager_S::CreateRequest service to create a distributed request object
return

```

#### **DCEObjRefT\* InRequestManager\_S::CreateRequest(DANmsg\* DANmsgPtr)**

```

Create an InRequest object
return

```

#### **int InRequestManager\_S::InRequestManager()**

```

BeginPDL

```

```

Call InRequestManager_S::RestoreRequestList to determine if checkpointed requests are avail-
able and to resubmit them as needed
Listen for request creation or cancellation requests
If (a creation request is received)
    Access MSS Configuration files to GetIngestPriority to get the priority for the specified request
    Call InRequestManager_S::SetRequestThresholds
    Create a pthread for the InRequest object
    Call InRequest::InRequest to create a new request
    Return to listening for a request
Elseif (StateChange request is received ...eg cancel suspend, resume, set priority)
    Call InRequestManager::ProcessRequestStateChange to process the request
    Return to listening for a request
Else
    Report an error to the operator and the log
endif
return
EndPDL

```

### **InRequestManager\_S::SetRequestThresholds**

```

BeginPDL
Access MSS configuration files to get external data provider_thresholds information
Access MSS configuration files to get systemThreshold information.
Access global current_provider_utilized (volume and total files)data for external data provider
Access global current_system_utilized (volume and total files)data for system
Calculate request allocation for volume and files
Return
EndPDL

```

### **InRequestManager::ProcessRequestStateChange(EctInt RequestID, RWString StateControlCommand)**

```

BeginPDL
This is routine is called by ProcessRequest when an RPC sent from InRequestController GUI task
State Control Commands include cancel, suspend, resume, recover, set priority. Note recover
would be sent from InReqeustManager::Recover function.
Begin PDL

```

Locate correct InRequest object ( .. currently how to do this is being investigated, multiple approaches are possible)

```

If request could not be located ... request may have just completed
    Query InRequestProcessHeader to determine current state of request
        If request has just completed naturally.
            Send informational message to the operator
            Log message locally

```

```

        return
    Endif

    else.. request not found in data base or found and still active (why is it not accessible then)
        Send error to operator indicating an invalid RequestID was sent
        Log error
        return
    Endif
Else ... request located
    Set myRequestState attribute in InRequest object... to new state
    This will signal InRequest of state change
EndPDL

```

### **int InRequestManager\_S::RestoreRequestList()**

```

BeginPDL
Call InRequestProcessHeader::SearchTable to determine whether request information is check-
pointed in an active state ("request created", "data transferred", "data preprocessed", "data in-
sert submitted", "data insert completed")
For each checkpointed request
    Access MSS configuration file to GetIngestPriority to get the priority for the specified request
    Create a pthread for the InRequest object
    Call InRequest::InRequest to create a new request
endfor
return
EndPDL

```

### **InRequestProcessData::GetGranuleStateForRecovery(EcTInt RequestID, EcTInt Gran- uleID, RWString ResultingAction)**

```

Begin PDL
This function is implemented as a Sybase stored proceedure.
    It determines if the granuleState is set to complete.
    If not complete, it will check to see if the granule has been previously restarted.
    If previously restarted set all file completion codes to RecoveryCausedAbortToRepeat to
indicated "twice caused ECS abort"
    Finally it sets myRetryFlag to 1 to flag this granule as having been previously restarted.
If Failed to find granule in database
    set error return to Failed
    Log Error
    retrun
Endif
If not complete and myRetryFlag was 0 (not preeviously restarted)
    Set ResultingAction to "Continue"

```

```

ElseIf myRetryFlag was 1
    Log informational message that this granule will not be restarted a second time
    Set ResultingAction to "SkipGranule"
Endif

Return
EndPDL

```

### **InRequestProcessHeader::UpdateStateForStateChange (EcTInt RequestID, RWstring StateControlCommand, EcTInt NewPriority, RWstring PreviousState)**

Note: this query should call a stored procedure, which locks out access to the request header record.

If the request is still in session mode, the stored procedure will mark it suspended or cancelled (as requested). The session software will check to see if it has been cancelled or suspended before sending it on to the request manager for processing

Possible RequestState values in InRequestProcessHeader

```

Session (still in session mode)Active (Managed by RequestManager)
    SuspendingActive
    Suspended
    CancellingActive
    Cancelled
    RetainedWithErrors
    Resuming

```

Possible Control Commands include

```

Suspend
Resume
Cancel
Mark Suspended
Mark Active (Resumed)
Mark Cancelled

```

C Procedure::

Call stored procedure

If error return

Log error

Return error and previous state to caller (differentiating between Sybase/stored procedure

errors and Invalid Control Request Errors

else

Send back success status to caller

Endif

Stored Procedure::

Lock access to request header record

Obtain previous state

If control command is not to mark active or mark cancelled or mark suspended and

```

        previous state control command has not completed, i.e. the request status is
pending, resuming, cancelling)
        Set failure status ... previous control command is not complete
        Pass back previous setting
        Return
    ElseIf
        If control command is to suspend
            if already suspended or suspending or cancelling or cancelled or Retained-
WithErrors
                return error and previous state
            else
                if previous state is session
                    set state to suspended
                else if previous state is active
                    set state to suspending ... InRequest will call to mark it sus-
pending
                else
                    return error indicating invalid previous state
                endif
            endif
        ElseIf control command is to mark suspended/mark active/ mark cancelled
            ... InRequest will issue these request (not the GUI)
            If previous state is not marked suspending/ resuming/ cancelling (corre-
spondingly)
                return error
            else
                set state to suspended/active/cancelled
            endif
        ElseIf control command is to cancel
            If previous state is suspended or retainedOnError or Session
                set state to cancelled
                call stored procedure to cancel the request
            elseif previous state is cancelled
                return error
            elseif previous state is active
                set state to cancelling
            else
                return error
            endif
        ElseIf control command is to resume
            If previous state is suspended or retainedOnError
                set state to resuming
            else
                return error

```

```

                endif
            Else ... not valid Control Command
                return error
            Endif
        Return
    End PDL

```

### **InServer::InServer()**

Initialize the SessionCount to zero.

### **InServer::StartServer()**

Setup Ingest RPC Server:

```

    Create instance of the InServer object class
    (UUID can be automatically created by constructor or passed to
    the constructor)
    Register Object (Place info about the object in the private state)
    Listen and wait for client request
    return

```

### **InSession::InSession()**

Initialize the default constructor prototype

### **InSession::InSession(char \*GatewayBH, char \*ClientId, int SessionId)**

Initialize the alternate constructor

### **InSession::~~InSession()**

Delete any dynamically allocated memory

### **InSession::InitSessServer(char \*SessGWBH)**

Setup Ingest RPC Session Server:

```

    Create instance of the InSession object class
    (UUID can be automatically created by constructor or passed to
    the constructor)
    Register Object (Place info about the object in the private state)
    Listen and wait for client request(DAN, DDA...)
    return

```

### **InSession::ProcessRequest(DANmsg \*DAN, DAAMsg \*DAA)**

BeginPDL

Construct an InDAN object

Validate

InDAN::CheckpointRequest

From MSS configuration object::GetPriority to get the current request priority

Call InExternalDataProviderThreshold::GetState to get the current state of the external Data Provider

If State is equal to suspend

Set DAA status to indicate the request has failed (since ECS has suspended the External DataProviders Session)

Set request failure status

return

Else

Send request to InRequestManager

Endif

Return

EndPDL

### **InSessionInfo::AddSession(char \*NewClient, int SessPID )**

Append Client and Session PID Information to file

Increment number of sessions open

return

### **InSessionInfo::DeleteSession(int SessionID)**

set FOUND to false

open Ingest Session file

while not found or end-of-file not reached

read a record

match the session id field in the record

If Session ID found

delete the record from the file

set FOUND to true

decrement number of sessions active

log client/session deletion

else

read next record

endif

endwhile

```
If not found
    log error - trying to delete non-existent session
endif
return (found)
```

### **InSessionInfo::ListSessions(void)**

```
Open Session List file
for each session in the session file
    print SessionId and ClientID
end for
close file
return
```

### **InSessionInfo::SearchSession (char \*CID )**

```
set FOUND to false
open Ingest Session file
while not found or end-of-file not reached
    read a record
    match the ClientID field in the record
    If Client ID found
        set FOUND to true
    endif
endwhile
close file
return FOUND
```

### **InSessionInfo::SearchSession ( int SessionId)**

```
set FOUND to false
open Ingest Session file
while not found or end-of-file not reached
    read a record
    match the session id field in the record
    If Session ID found
        set FOUND to true
    endif
endwhile
close file
return FOUND
```

### **int InShortDAA::FillDAA(int DAASStatus, int DANSeqNo)**

```

If (DANSeqNo is greater than 0)
    Assign DANSeqNo to myShortDAA.DANSequenceNum
    Assign DAAStatus to myShortDAA.Disposition
Else
    Write invalid DAN Sequence Number into the event log
End if
Assign the byte size of ShortDAAMsg structure to ShortDAAMsgLen
Assign the least significant 3 byte of ShortDAAMsgLen
    to myShortDAA.MsgLength
return

```

### **int InShortDAA::GetDAA(ShortDAAMsg \*DAAMsg, int DAAMsgLen)**

```

Assign myShortDAA to DAAMsg
Call InMessage::GetMsgLength() and assign to DAAMsgLen
If (DAAMsgLen is <= zero)
    write Empty DAA data message into the event log
Endif
return

```

### **InShortDAA::InShortDAA()**

```

Assign the least significant byte of InCShortDAAType to myShortDAA.MsgType
Return

```

### **InShortDDN::FillDDN(int DDNStatus, int DANSeqNo)**

```

If (DANSeqNo is greater than 0)
    Assign DANSeqNo to myShortDDN.DANSequenceNum
    Assign DDNStatus to myShortDDN.Disposition
Else
    write invalid DAN Sequence Number into the event log
End if
Assign the byte size of ShortDDNmsg structure to ShortDDNmsgLen
Assign the least significant 3 byte of ShortDDNmsgLen
    to myShortDDN.MsgLength
return

```

### **InShortDDN::GetDDN(ShortDDNmsg \*DDNmsg, int DDNmsgLen)**

```

Assign myShortDDN to DDNmsg

```

```
Call InMessage::GetMsgLength() and assign to DDNmsgLen
If (DDNmsgLe is <= zero)
    write Empty DAA data message into the event log
Endif
return
```

### **InShortDDN::InShortDDN()**

```
Assign the lease significant byte of InCShortDDNType to myShortDDN.MsgType
Return
```

The Ingest Program Design Language (PDL) for non-trivial operations is included in this appendix. The PDL is sorted by object class and by object class operation. C++ syntax is used to identify object class operations (e.g., InBOMetadata::Preprocess indicates the Preprocess operation of the InBOMetadata object class). The PDL follows the standards set up in the PDL Program Instruction addendum to the Software Development Plan.

## Appendix C. Ingest Recovery Analysis

The following table lists major Ingest subsystem problem categories, their impact and consequences, dependencies on other servers for recovery, the Ingest recovery action, and any required user or operator action.

**Table C-1. Ingest Problem Category, Problem Consequences, and Recovery Actions (1 of 4)**

<b>Problem Category</b>	<b>Impact/ Consequences</b>	<b>Dependency</b>	<b>Ingest Recovery Action</b>	<b>User/ Operator Action</b>
Ingest Gateway client unavailable when Ingest server transmitting Data Availability Ack (DAA)	Transparent to Ingest other than delay in completion of DAA delivery	Server Request Framework (SRF) delivers DAA after Ingest Gateway recovers.	Ingest retries an operator tunable number of times, then notifies operator and logs error.	None
Ingest Gateway client unavailable when Ingest server transmitting Data Delivery Notice (DDN)	Transparent to Ingest other than delay in completion of DDN delivery	Server Request Framework (SRF) delivers DDN after Ingest Gateway recovers.	Ingest retries an operator tunable number of times, then notifies operator and logs error.	None
Data Server unavailable when insert service ack or callback expected	Transparent to Ingest other than delay in completion of data granule insert	Server Request Framework (SRF) delivers insert ack or callback after Data Server recovers.	None	Operator: detects timeout of request if insert is not performed before expiration date/time; manual capability to cancel request
Ingest Polling Session fails	Transparent to Ingest; no impact to other ingest requests	Detection of process failure by MSS agent	Periodic polling re-initiated after MSS Process Framework (PF) restart	Operator: receives alert of failed process; restarts process after diagnosis by means of PF
Ingest Session Manager (Server/ Session objects) fails	Delay of request completion until Session Manager recovers; no new ingest requests accepted until after recovery	Detection of process failure by MSS agent	Session Manager recovers from checkpointed information after MSS Process Framework (PF) restart	Operator: receives alert of failed process; restarts process after diagnosis by means of PF

**Table C-1. Ingest Problem Category, Problem Consequences, and Recovery Actions (2 of 4)**

<b>Problem Category</b>	<b>Impact/ Consequences</b>	<b>Dependency</b>	<b>Ingest Recovery Action</b>	<b>User/ Operator Action</b>
Ingest Request Manager (Request Manager/ Request objects) fails	Delay of all request completions until Request Manager recovers	Detection of process failure by MSS agent	Request Manager recovers from checkpointed information after MSS Process Framework (PF) restart	Operator: receives alert of failed process; restarts process after diagnosis by means of PF
Ingest Granule Server (Granule Server/ Granule objects) fails	Delay of granule completion (and related request completion) until Granule Manager recovers	Detection of process failure by MSS agent	Granule Manager recovers from checkpointed information after MSS Process Framework (PF) restart	Operator: receives alert of failed process; restarts process after diagnosis by means of PF
UNIX signals (e.g., numeric fault)	Abort of affected Ingest process	Detection of process failure by MSS agent	Ingest server recovers from checkpointed information after MSS Process Framework (PF) restart	Operator: receives alert of failed process; Sustaining Engineering diagnoses using Ingest event log and UNIX core dump files; restarts process after diagnosis by means of PF
Ingest processor failure	Abort of Ingest processes running on processor	Detection of processor failure by MSS agent	Ingest processes recover from checkpointed information (as described above) after MSS Process Framework (PF) restart	Operator: receives alert of failed process; restarts process after diagnosis by means of PF
ftp fault (polling) (NESDIS SAA, DAO, DAACs, EDOS, other data providers)	No recovery of current request; polling is retried at next interval; no other Ingest request affected	None	None	Operator: receives alert of ftp failure; diagnoses problem and communicates with external data provider
ftp fault (DAN ingest) (TSDIS, SDPF, Landsat-7 LPS, other data providers)	No recovery of current request; no other Ingest request affected	None	Reports failure to external data provider by means of DDN	Operator: receives alert of ftp failure; diagnoses problem and communicates with external data provider

**Table C-1. Ingest Problem Category, Problem Consequences, and Recovery Actions (3 of 4)**

<b>Problem Category</b>	<b>Impact/ Consequences</b>	<b>Dependency</b>	<b>Ingest Recovery Action</b>	<b>User/ Operator Action</b>
Sybase fault: templates needed for preprocessing granule cannot be accessed	No recovery of current request; subsequent requests fail if template information not available	None--assume Sybase switchover to secondary server fails	None--this failure occurs very rarely	Operator: receives alert of DB access failure; diagnoses problem and restarts Sybase as needed
Sybase fault: DB update of Ingest History summary log or checkpointing log could not be completed	Transparent to Ingest; checkpointing and summary information not retained until recovery performed	None--assume Sybase switchover to secondary server fails	None--this failure occurs very rarely	Operator: receives alert of DB access failure; diagnoses problem and restarts Sybase as needed
Loss of Sybase transaction log	No impact to ongoing Ingest requests	Sybase switchover to mirrored transaction log disk	None	Operator: receives Sybase notification of switchover
Metadata validation failure with abort (e.g. Constraints fault, subscription out of range)	No impact to ongoing ingest requests	None	Reports failure of data granule to data provider	Operator: receives alert of metadata validation failure; resolves with data provider
Unable to allocate disk space	Delay of current request until disk space available; no impact to other requests	Data Server queues requests for disk space and satisfies when disk space is available	None	Operator: reviews status of disk space allocations
Disk crash (working storage disk managed by Data Server and used by Ingest for data granule preprocessing)	Failure of ongoing granule processing occurring on the failed disk (note: this failure occurs rarely, since RAID-5 is used for working storage)	MSS agent detects and reports on disk failure	Reports failure to external data provider by means of DDN	Operator: receives notification of disk failure; uses Data Server services to diagnose and fix disk failure
Ingest media or media peripheral failure (hardware controlled by Data Server)	Failure of affected ingest request; no impact to other ongoing requests	None	None	Operator: receives notification of media or media peripheral failure; uses Data Server services to diagnose and fix; resubmits media ingest request

**Table C-1. Ingest Problem Category, Problem Consequences, and Recovery Actions (4 of 4)**

<b>Problem Category</b>	<b>Impact/ Consequences</b>	<b>Dependency</b>	<b>Ingest Recovery Action</b>	<b>User/ Operator Action</b>
Unable to deliver email message	Transparent to Ingest; no impact to ongoing requests	ECS mail server queues email message for delivery	None	None
GUI - Invalid User I/F Screen Input	Transparent to Ingest	None	Rejects invalid input when detected (note: many fields provide a selection list of valid input)	User: reenters valid input

# Acronyms and Abbreviations

---

ADC	Affiliated Data Center
AM-1	EOS AM Project (morning spacecraft series)
APID	Application Identifier
ASCII	American Standard Code for Information Interchange
ASF	Alaska SAR Facility (DAAC)
ASTER	Advanced Spaceborne Thermal Emission and Reflection Radiometer (formerly ITIR)
ATL	Automated Tape Library
ATM	Asynchronous Transfer Mode
AVHRR	Advanced Very High-Resolution Radiometer
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CD-ROM	Compact Disk - Read Only Memory
CERES	Clouds and Earth's Radiant Energy System
CGI	Common Gateway Interface
CI	Configuration Item
CIESIN	Consortium for International Earth Science Information Network
COTS	Commercial-off-the-shelf
CPU	Central Processing Unit
CSC	Computer System Components
CSCI	Computer Software Configuration Item
CSMS	Communications and Systems Management Segment (ECS)
DAA	DAN Acknowledge
DAAC	Distributed Active Archive Center
DAN	Data Availability Notice
DAO	Data Assimilation Office
DBMS	Database Management System
DCE	Distributed Computing Environment (OSF)
DDA	Data Delivery Acknowledgment
DDN	Data Delivery Notice
DID	Data Ingest Distribution
DID	Data Item Description

DIPHW	Distribution and Ingest Peripheral Management HWCI
DPRHW	Data Repository HWCI
DPS	Data Processing Subsystem
DSS	Data Server Subsystem
EBnet	EOSDIS Backbone network
Ecom	EOSDIS communications system
ECS	EOSDIS Core System
EDC	EROS Data Center (DAAC)
EDOS	EOS Data and Operations System
EDR	Environmental Data Record
EGS	EOS Ground System
EOC	EOS Operations Center (ECS)
EOS	Earth Observing System
EOSDIS	Earth Observing System Data and Information System
FDDI	Fiber Distributed Data Interface
FDF	Flight Dynamics Facility
FNL	Final Analysis and Forecast System, Global Analysis
FSMS	File Storage Management System
FIP	File Transfer Protocol
GB	Gigabyte
GDAO	GSFC Data Assimilation Office
GPCP	Global Precipitation Climatology Project
GPI	GOES Precipitation Index
GRIB	Gridded Binary
GSFC	Goddard Space Flight Center
GUI	Graphic User Interface
GVTRMM	Ground Verification Tropical Rainfall Measuring Mission
H/K	Housekeeping
HDF	Hierarchical Data Format
HiPPI	High Performance Parallel Interface
HTML	Hyper-Text Markup Language
HTTP	Hypertext Transport Protocol
HWCI	Hardware Configuration Item
I/F	Interface
I/O	Input/Output

IAS	Image Assessment System (Landsat-7)
IAS	Instrument Activity Specification
ICD	Interface Control Document
ICLHW	Ingest Client HWCI
ID	Identification
IDR	Interim Design Review
IGS	International Ground Station
IP	International Partners
IR-1	Interim Release-1
IRD	Interface Requirements Document
ISSCP	International Satellite Cloud Climatology Project
JPL	Jet Propulsion Laboratory
L0	Level-0
LaRC	Langley Research Center (DAAC)
LIS	Lightning Imaging Sensor
LOM	Logical Object Model
Mb	Mega bit
MB	Mega byte
Mbps	Mega bits per second
MCF	Metadata Configuration File
MISR	Multi-Angle Imaging SpectroRadiometer
MOC	Mission Operations Center
MODIS	Moderate-Resolution Imaging SpectroRadiometer
MRF	Medium Range Forecast
MSFC	Marshall Space Flight Center
MSS	Management Subsystem
MTBF	Mean Time Between Failure
MTTR	Mean Time To Restore
NESDIS	National Environmental Satellite, Data, and Information Service (NOAA)
NMC	National Meteorological Center (NOAA)
NMC	Network Management Center
NOAA	National Oceanic and Atmospheric Administration
NOLAN	Nascom Operational Local Area Network
NSIDC	National Snow and Ice Data Center (DAAC)
OODCE	Object Oriented DCE

ORNL	Oak Ridge National Laboratory (DAAC)
PDL	Program Design Language
PDR	Preliminary Design Review
PDS	Production Data Set
POAM-II	Polar Ozone and Aerosol Experiment
PS	Project Scientist
PVL	Parameter Value Language
QA	Quality Assurance
RAID	Redundant Array of Inexpensive Disks
RID	Review Item Discrepancy
RMA	Reliability, Maintainability, Availability
RPC	Remote Procedure Call
SAGE II	Stratospheric Aerosol and Gas Experiment
SCF	Science Computing Facility
SCSI	Small Computer System Interface
SDPF	Sensor Data Processing Facility (GSFC)
SDPS	Science Data Processing Segment (ECS)
SDR	System Design Review
SDSRV	Science Data Server CSCI
SFDU	Standard Format Data Unit
SMC	System Management Center (ECS)
SMP	Symmetric Multi-Processing
SNMP	Simple Network Management Protocol
SSM/I	Special Sensor for Microwave/Imaging
STMGT	Storage Resource Management
TBD	To Be Determined
TBR	To Be Replaced
HDF	Hierarchical Data Format
TBS	To Be Supplied
TCP/IP	Transmission Control Protocol/Internet Protocol
TDRSS	Tracking and Data Relay Satellite System
TOMS	Total Ozone Mapping Spectrometer
TRMM	Tropical Rainfall Measuring Mission (joint US-Japan)
TSDIS	TRMM Science Data and Information System
UR	Universal Reference

UUID	Universal Unique Identifier
W/S	Workstation
WKSHW	Working Storage HWCI
WWW	World Wide Web

This page intentionally left blank.

# Glossary

---

advertisement	A text description that announces the availability of ECS data or services to ECS users.
advertising service	Through the advertising service, users can search and query descriptions of the data and services available in the network. This data is called advertisements. It is prepared by the data and/or service providers.
affiliated data center (ADC)	A facility not funded by NASA that processes, archives, and distributes Earth science data useful for global change research, with which a working agreement has been negotiated by the EOS program. The agreement provides for the establishment of the degree of connectivity and interoperability between EOSDIS and the ADC needed to meet the specific data access requirements involved in a manner consistent and compatible with EOSDIS services. Such data-related services to be provided to EOSDIS by the ADC can vary considerably for each specific case.
ancillary data	Data other than instrument data required to perform an instrument's data processing. They include orbit data, attitude data, time information, spacecraft engineering data, calibration data, data quality information, and data from other instruments.
application identifier (APID)	The number assigned by spacecraft mission management that represents the on-board application that generated the telemetry data.
application software	Programs designed for specific functions, such as payroll, accounts payable, inventory control, or property management, generally consisting of source code and object code databases, procedures, and documentation
archive tape library	Archive robotics unit
authorized user	see user, authorized
availability	A measure of the degree to which an item is in an operable and committable state at the start of a "mission" (a requirement to perform its function) when the "mission" is called for an unknown (random) time. (Mathematically, operational availability is defined as the mean time between failures divided by the sum of the mean time between failures and the mean down time [before restoration of function].)

baseline	Identification and control of the configuration of software (i.e. selected software work products and their descriptions) at given points in time.
binary file	A data file whose contents are in binary form (i.e., not encoded)
browse data product	Subsets of a larger data set, other than the directory and guide, generated for the purpose of allowing rapid interrogation (i.e., browse) of the larger data set by a potential user. For example, the browse product for an image data set with multiple spectral bands and moderate spatial resolution might be an image in two spectral channels, at a degraded spatial resolution. The form of browse data is generally unique for each type of data set and depends on the nature of the data and the criteria used for data selection within the relevant scientific disciplines.
calibration	The collection of data required to perform calibration of the instrument science data, instrument engineering data, and the spacecraft engineering data. It includes pre-flight calibration measurements, in-flight calibrator measurements, calibration equation coefficients derived from calibration software routines, and ground truth data that are to be used in the data calibration processing routine.
CCSDS recommendations	Recommendations for spacecraft telemetry and telecommand packet format and protocol made by the Consultative Committee for Space Data Systems.
client	A software component that sends or issues service requests to ECS servers or service providers; a requester of service.
client session	see SESSION
commercial off the shelf (COTS)	COTS is a product, such as an item, material, software, component, subsystem, or system, sold or traded to the general public in the course of normal business operations at prices based on established catalog or market prices (see FAR 15.804-3(c) for explanation of terms.
component	The next lower functional subdivision below "subsystem" in the ECS functional hierarchy.
computer software component (CSC)	A distinct part of a computer software configuration item. CSCs may be further decomposed into other CSCs and computer software units.
computer software configuration item (CSCI)	A configuration item comprised of computer software components and computer software units.

configuration	The functional and physical characteristics of hardware, firmware, software or a combination thereof as set forth in technical document and achieved in a product.
configuration item (CI)	An aggregation of hardware, firmware, software or any of its discrete portions, which satisfies an end use function and is designated for configuration management.
Critical Design Review (CDR)	A detailed review of the element/segment-level design, including such details as program design language for key software modules, and element interfaces associated with a release.
DAAC	see Distributed Active Archive Center
DAAC-unique	Functions and capabilities provided by the DAAC beyond those provided by the core system. The functions will be integrated with ECS via APIs for other similar mechanisms. Examples of DAAC-unique functions include visualization, specialized interfaces, and data set-unique functionality.
Data Archive And Distribution System (DADS)	Included in each DAAC and responsible for archiving and distribution of EOS data and information.
data availability acknowledgment	Status return when a data availability notice cannot be satisfied (e.g., due to a validation error or transmission error).
data availability notice	Notice from a client of data available for ingest.
data availability schedule	Data availability schedule is a schedule indicating the times at which specific data sets will be available from remote DADS, EDOS, the international partners, the ADCs, and other data centers for ingestion by the collocated DADS. The schedules are received directly by the PGS.
data center	A facility storing, maintaining, and making available data sets for expected use in ongoing and/or future activities. Data centers provide selection and replication of data and needed documentation and, often, the generation of user tailored data products.
data ingest request	Request to ingest data.

data product	<p>Data products consist of Level 0 data or Level 1 through Level 4 data products obtained by the PGS from the collocated DADS. These represent the primary input to the product generation process.</p> <p>A collection (1 or more) of parameters packaged with associated ancillary and labeling data, uniformly processed and formatted. Typically uniform temporal and spatial resolution. (Often the collection of data distributed by a data center or subsetted by a data center for distribution.) There are two types of data products:</p> <p>a.Standard: A data product produced at a DAAC by a community consensus algorithm. Typically produced for a wide community. May be produced routinely or on-demand.</p>
data product levels	<p>Raw data--Data in their original packets, as received from the observer, unprocessed by EDOS.</p> <ul style="list-style-type: none"> <li>• Level 0--Raw instrument data at original resolution, time ordered, with duplicate packets removed.</li> <li>• Level 1A--Reconstructed unprocessed instrument data at full resolution, time referenced, and annotated with ancillary information, including radiometric and geometric calibration coefficients and georeferencing parameters (i.e. platform ephemeris) computed and appended, but not applied to Level 0 data.</li> <li>• Level 1B--Radiometrically corrected and geolocated Level 1A data that have been processed to sensor units.</li> <li>• Level 2--Derived geophysical parameters at the same resolution and location as the Level 1 data.</li> </ul>
data server	<p>Either the data server subsystem as a whole, or a specific instance of a data server. A data server is a (hardware/software) entity that accepts, stores, and distributes EOS (and other) data, for both other subsystems within ECS and external users.</p>
data server insert request	<p>Request to insert data into a data server.</p>
data set	<p>A logically meaningful grouping or collection of similar or related data.</p>
data type	<p>A particular type of data handled by a particular data server. An example of a data type might be MODIS Level 1a products, etc.</p>

data type taxonomy	A classification of earth science and related data into types.
definitive attitude data	Down-linked attitude data received with Level 0 data.
definitive orbit data	Down-linked orbit (ephemeris) data received with level 0 data.
delivered algorithm packages	The full content of data and information delivered by a data producer during the process of standard product Algorithm Integration & Test, including all elements defined as minimum content within Volume 4 of the Science User's Guide, available at PDR.
Distributed Active Archive Center (DAAC)	An EOSDIS facility which generates, archives, and distributes EOS Standard Products and related information for the duration of the EOS mission. An EOSDIS DAAC is managed by an institution such as a NASA field center or a university, per agreement with NASA. Each DAAC contains functional elements for processing data (the PGS), for archiving and disseminating data (the DADS), and for user services and information management (elements of the IMS).
	ASF -- Alaska SAR Facility
	EDC -- EROS Data Center
	GSFC -- Goddard Space Flight Center
	JPL -- Jet Propulsion Laboratory
EDOS data unit (EDU)	The message packet generated by EDOS that contains the reconstructed spacecraft telemetry packet.
engineering data	<p>All data available on-board about health, safety, environment, or status of the spacecraft and instruments.</p> <ul style="list-style-type: none"> <li>• housekeeping data: The subset of engineering data required for mission and science operations. These include health and safety, ephemeris, and other required environmental parameters.</li> <li>• instrument engineering data: All non-science data provided by the instrument.</li> <li>• platform engineering data: The subset of engineering data from platform sensor measurements and on-board computations.</li> <li>• spacecraft engineering data: The subset of engineering data from spacecraft sensor measurements and on-board computations.</li> </ul>

EOS Data and Operations System (EDOS) production data set	<p>Data sets generated by EDOS using raw instrument or spacecraft packets with space-to-ground transmission artifacts removed, in time order, with duplicate data removed, and with quality/ accounting (Q/A) metadata appended. Time span, or number of packets, encompassed in a single data set are specified by the recipient of the data. These data sets are equivalent to Level 0 data formatted with Q/A metadata.</p> <p>For EOS, the data sets are composed of: instrument science packets, instrument engineering packets, spacecraft housekeeping packets, or onboard ancillary packets with quality and accounting information from each individual packet and the data set itself and with essential formatting information for unambiguous identification and subsequent processing.</p>
ephemeris data	See "orbit data"
external data provider	An external data source providing data to be ingested in SDPS.
format	Format of data -- ASCII, binary, etc.
granule	The smallest aggregation of data that is independently managed (i.e., described, inventoried, retrievable). Granules may be managed as logical granules and/or physical granules.
granule location	The name of the product where this granule is located.
hardware	That combination of subcontracted, COTS, and government furnished equipment (e.g., cables and computing machines) that are the platforms for software.
hardware configuration item (HWCI)	A configuration item comprised of hardware components.
HDF file	A data file whose format follows the NCSA Hierarchical Data Format standard, as well as ECS-developed extensions thereto.
I/O access	A read or write by a process to a data file.
ingest status request	Request for status on a data ingest request.
insert request	Request to insert data into the archive.
interface classes	The interfaces offered by a class of objects or object collections. User, for example, in the context of Service Classes to denote the collection of interfaces supported by this service class.
interface definition language (IDL)	IDL provides uniform semantics for all interfaces.

interface(s)	The functional and physical characteristics required to exist at a common boundary.
maintainability	The measure of the ability of an item to be retained in or restored to a specified condition when maintenance is performed by personnel having specified skill levels, using prescribed procedures and resources, at each prescribed level of maintenance and repair. (The probability that maintenance, both corrective and preventive, can be performed in a specified amount of time using a specified set of prescribed procedures and resources expressed as MTTR). Maintainability is the function of design.
mean down time (MDT)	Sum of the mean time to repair MTTR, plus the average administrative logistic delay times.
mean time between failure (MTBF)	The reliability result of the reciprocal of a failure rate that predicts the average number of hours that an item, assembly or piece part will operate within specific design parameters. (MTBF=1/(l) failure rate; (l) failure rate = # of failures/ operating time.
mean time to repair (MTTR)	The mean time required to perform corrective maintenance to restore a system/equipment to operate within design parameters. It is a basic measure of maintainability: The sum of corrective maintenance times at any specific level of repair, divided by the total number of failures within an item repaired at that level, during a particular interval under stated conditions.
metadata	Information about data sets which is provided to the ECS by the data supplier or the generating algorithm and which provides a description of the content, format, and utility of the data set. Metadata may be used to select data for a particular scientific investigation. It is “data about data” used to facilitate database searches. Types of metadata include: product metadata (data describing a particular product, such as when it was generated, etc.) and algorithm metadata (data describing science software)
object	Identifiable encapsulated entities providing one or more services that clients can request. Objects are created and destroyed as a result of object requests. Objects are identified by client via unique reference.
object implementation	Code and data that realizes target object's behavior.
operations personnel	Same as operations staff.

operations staff	Generic term for personnel who have the responsibility to operate, monitor, and control SDPS. Also can be, one of the DAAC operations staff assigned to the ingest or data server subsystems, i.e., Data Archive Analyst, Data Ingest Technician, Data Distribution Technician, Data Base Administrator, etc.
orbit data	Data that represent spacecraft locations. Orbit (or ephemeris) data include: Geodetic latitude, longitude and height above an adopted reference ellipsoid (or distance from the center of mass of the Earth); a corresponding statement about the accuracy of the position and the corresponding time of the position (including the time system); some accuracy requirements may be hundreds of meters while other may be a few centimeters.
p = v metadata	Label = value where label is a field name and value is either a single value or list of values.
Parameter Value Language	Also expressed as “PVL”. A P=V metadata language used to package metadata for storage in the Data Server subsystem.
Preliminary Design Review (PDR)	PDR is held for each ECS Segment. The PDR addresses the design of the segment-level capabilities and element interfaces through all ECS releases. The PDR also addresses prototyping results and how the results of both Contractor and Government prototyping efforts, studies, and user experience with EOSDIS Version 0 have been incorporated into the ECS design for each respective Segment.
process	An executing program.
PVL	See Parameter Value Language
quick-look data	Data received during one TDRSS contact period which have been processed to Level 0 (to the extent possible for data from a single contact).
reliability	Reliability is the function of design. It is the probability that system/equipment will operate within design parameters under stated conditions, for a specified interval expressed as MTBF.
report	Documentation of some automated (such as standards checking ) or manual (such as evaluation of a science software delivery) activity.
requirement	A statement to which the developed system must comply. Varieties of requirements: Levels 2, 3, 4; performance, functional, design, interface.

requirements traceability	<p>There are three recognized levels of requirements on the ECS Project:</p> <ul style="list-style-type: none"> <li>• ESDIS (Level 2)</li> <li>• ECS System (Level 3)</li> <li>• ECS Detailed Subsystem (Level 4)</li> </ul> <p>Traceability is the verification and validation of the parents and children of ECS Levels 2,3,4 requirements down to release and subsystem levels. Analysis is done by the ECS Project System and Subsystem engineering.</p>
reusable software	Software developed in response to the requirements for one application that can be used, in whole or in part, to satisfy the requirements of another application.
scenario	A description of the operation of the system in user's terminology including a description of the output response for a given set of input stimuli. Scenarios are used to define operations concepts.
science user	A user the SDPS from the scientist community or other user community that originates service requests.
SDP Toolkit	A set of SDPS-standard API between science algorithms and the process execution service for status reporting and process control
server	A software component that receives and executes service requests (e.g., the LIM, the DIM, the data server, the PLANG CI).
service	A grouping of functional requirements as listed in a specification. For example, in the Level 3 requirements, IMS "services" are System Access, Information Search, etc.

session	<p>The logical context assigned to a user or a client in which a set of service requests are performed. Sessions associate and manage the resources and results sets that are allocated and generated as a result of the processing of service requests. A session retains information associated with the execution of service requests so that it is accessible to subsequent service requests. Service requests may utilize resources and results sets allocated and produced by other service requests belonging to the same session. Service requests issued in the context of one session cannot utilize the resources managed by another session. There are two kinds of sessions, client sessions and user sessions.</p> <p>Sessions have the following states:</p> <p>a.Active: The session is established and will allow service requests to allocate and access session resources.</p>
session, client	<p>A client session supports interactions between a client and a server. Client sessions associate and manage the resources and results sets that are allocated and generated by the server.</p>
simulated data	...same as test data
status	<p>Status is information regarding schedules, hardware and software configuration, exception conditions, or processing performance. This information is exchanged with the DADS, and is provided to the system management center (SSMC). The SSMC may also receive information regarding schedule conflicts that have not been resolved with the IMS.</p>
status request	Request for status of archive insert and retrieval requests (also need this for ingest and distribution).
universal reference	A uniform model for referencing objects throughout SDPS which each SDPS service will understand and support.

user

- Any person accessing the EOSDIS.
- Authorized users are users who have viable EOSDIS accounts, and who may therefore make EOSDIS data requests. These users may be affiliated or unaffiliated. Affiliated users are those who are sponsored by one of the parties to the Earth Observations-International Coordination Working Group (EO-ICWG) data policy. Each party is responsible for ensuring that all its affiliated users comply with the EO-ICWG data policy. Use of data by affiliated users is classified in one of three categories, defined in the EO-ICWG data policy:

World Wide Web  
browser

+ Research Use: A study or an investigation in which the user affirms (1) the aim is to establish facts or principles; (2) the data will not be sold or reproduced or provided to anyone not covered by this or another valid affirmation; (3) the results of the research will be submitted for publication in the scientific literature (local or remote) that allows a user to Access the WWW either textually or graphically. WWW is a mechanism for connecting Internet via a set of hypertext documents.

This page intentionally left blank.